

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПІДТРИМКИ ПЛАНУВАННЯ ДІЯЛЬНОСТІ ТУРИСТИЧНОЇ
КОМПАНІЇ. RESEARCH AND SOFTWARE IMPLEMENTATION OF
METHODS FOR PLANNING THE TRAVEL AGENCY ACTIVITY

Виконав(ла): студент(ка) 2 курсу, групи КНТ-142м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГОРОВЕНКО Д.Р.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2023_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГОРОВЕНКА Дмитра Романовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підтримки планування діяльності туристичної компанії. Research and Software Implementation of Methods for Planning the Travel Agency Activity

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) _____ рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Реалізація програми. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-4 тижні	Розділ 2
4	Розробка архітектури програми	5 тиждень	Розділ 3
5	Розробка програми	6-7 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	8-9 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї	10 тиждень	Додатки
8	Нормоконтроль та рецензування	11 тиждень	
9	Захист роботи	12 тиждень	

Студент(ка)

_____ Дмитро ГОРОВЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Степан СКРУПСЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
109 с., 26 рис., 4 табл., 3 дод., 12 джерел.

ТУРИЗМ, ТУРИСТИЧНА КОМПАНІЯ, ТУРИСТИЧНИЙ ПАКЕТ,
ВЕБЗАСТОСУНОК, МАШИННЕ НАВЧАННЯ.

Об'єкт роботи – процес забезпечення діяльності туристичної компанії.
Предмет роботи – методи підтримки планування діяльності туристичної компанії. Мета роботи – розробити програмне забезпечення для підтримки діяльності туристичної компанії щодо прийняття обґрунтованих рішень на основі моніторингу основної діяльності компанії і програмного визначення вартості туристичних пакетів.

Матеріали, методи та технічні засоби: мова Python, фреймворк Django, система PostgreSQL.

Результати. Програма, яку було розроблено, виконує підтримку забезпечення діяльності туристичної компанії через надання засобів для створення туристичних пакетів, визначення дат подорожей, варіантів вартості, визначення показників, отриманих від реалізації туристичного пакета, використання засобів прогнозування, забезпечення зручності пошуку користувачами як через надання інструментів пошуку різних даних, так і через надання реакцій різного типу і збереження в закладках для подальшого повернення.

Висновки. Наукова новизна роботи полягає в запропонованому методі визначення вартості туристичного пакета, який визначає обчислення вхідних ознак моделі на основі характеристик туристичного пакета, виконує створення моделі на основі екстремального градієнтного підсилювання для навчання ансамблів дерев рішень.

Галузь використання. Туризм, туристичні компанії.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 109 pages, 26 figures, 4 tables, 3 appendixes, 12 sources.

TOURISM, TRAVEL AGENCY, TOURIST PACKAGE, WEBAPPLICATION, MACHINE LEARNING.

The object of the work is the process of ensuring the activities of a travel company. The subject of the work is the methods of supporting the planning of the travel company's activities. The purpose of the work is to develop software to support the activity of a travel company in making informed decisions based on the monitoring of the company's main activities and the software determination of the price of travel packages.

Materials, methods and technical means: Python language, Django framework, PostgreSQL system.

The results. The program that was developed supports the operation of a travel company by providing tools for creating travel packages, determining travel dates, cost options, determining indicators obtained from the implementation of a travel package, using forecasting tools, ensuring ease of search by users, as well as by providing search tools for various data objects, as well as by providing reactions of various types and saving them in bookmarks for later return.

Conclusions. The scientific novelty of the work is in the proposed method of determining the cost of a tourist package, which determines the calculation of the input features of the model based on the characteristics of the tourist package, performs the creation of a model based on extreme gradient boosting for training of ensembles of decision trees.

Field of use. Tourism, travel companies.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Програмні інструменти для забезпечення роботи туристичної компанії.....	10
1.2 Порівняння розроблюваної програми з існуючими	13
1.3 Висновки за розділом 1	14
2 Матеріали і методи.....	15
2.1 Використання методів прогнозування для розв’язання задачі визначення вартості туристичного пакета	15
2.2 Метод визначення вартості туристичного пакета	17
2.3 Висновки за розділом 2	19
3 Розробка архітектури програми.....	20
3.1 Визначення сценаріїв використання програми користувачами	20
3.2 Рішення щодо вибору засобів розробки	23
3.3 Визначення структури даних у базі даних	24
3.4 Висновки за розділом 3	39
4 Реалізація програми	40
4.1 Визначення послідовності роботи з програмою	40
4.2 Особливості реалізації програми.....	41
4.3 Висновки за розділом 4	60
5 Експлуатація, тестування та експериментальне дослідження програми	61
5.1 Призначення програми	61
5.2 Виконання програми.....	61
5.3 Експериментальне дослідження	66
5.4 Висновки за розділом 5	70
Висновки	71
Перелік джерел посилання	72

Додаток А Технічне завдання	74
Додаток Б Текст програми	79
Додаток В Слайди презентації.....	102

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MAE – Mean Absolute Error;

MVT – Model View Template;

БД – база даних.

ВСТУП

За поточними прогнозами очікується, що глобальний туристичний ринок, обсяг якого у 2022 році був оцінений у 10,5 трильйонів доларів США, ще додатково буде збільшуватися на 5 % щорічно протягом наступного десятиріччя, що в підсумку має призвести до досягнення обсягу в 17,1 трильйони доларів США до кінця прогнозованого 2032 року [1]. Таким чином, незважаючи на поточний величезний обсяг туристичного ринку, в подальшому він буде ще збільшуватися, а це вказує на те, що є простір для розвитку і нових рішень, виходу на ринок.

Туристичні компанії є одними з активних рушіїв економічних процесів, а тому і програмне забезпечення, яке направлене на підтримку роботи цих компаній, має бути достатньо складним, тобто містити велику кількість інструментів, для того щоби підвищити зручність роботи з даними, які генеруються під час роботи туристичної компанії, отримати з них, інформацію, що може стати підґрунтям для прийняття рішень. Наявності лише форм для створення замовлень недостатньо для забезпечення ефективного набору інструментів за умов роботи в конкурентному середовищі. Тому розробка в цій роботі повинна враховувати і ґрунтуватися саме на цьому підході. Розроблення такого інструменту, який буде інтегрувати до своєї структури інструменти прогнозування для визначення показників, пов'язаних з туристичними пакетами, дозволить отримати інформацію, необхідну для прийняття рішень щодо подальшого розвитку окремих туристичних пакетів, взаємодії з різними компаніями-перевізниками, що буде сприяти в підсумку підвищенню ефективності рішень, які приймаються відповідною особою, що відповідальна за такі рішення в туристичній компанії.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Програмні інструменти для забезпечення роботи туристичної компанії

Однією з найвідоміших у світі глобальних туристичних компаній є TUI (рис. 1.1) [2]. Вебсайт дозволяє виконувати пошук туристичних пакетів, групуючи їх різноманітним чином, дозволяючи як виконувати пошук, так і переходити до сторінок з відповідними пакетами, представленими за групами, для чого виділені окремі розділи меню. Також можна переглядати рейси (польоти), місця призначення, готелі. За кожним готелем, наприклад, надається інформація про нього, інформація про кімнати, надається можливість виконати бронювання місць.

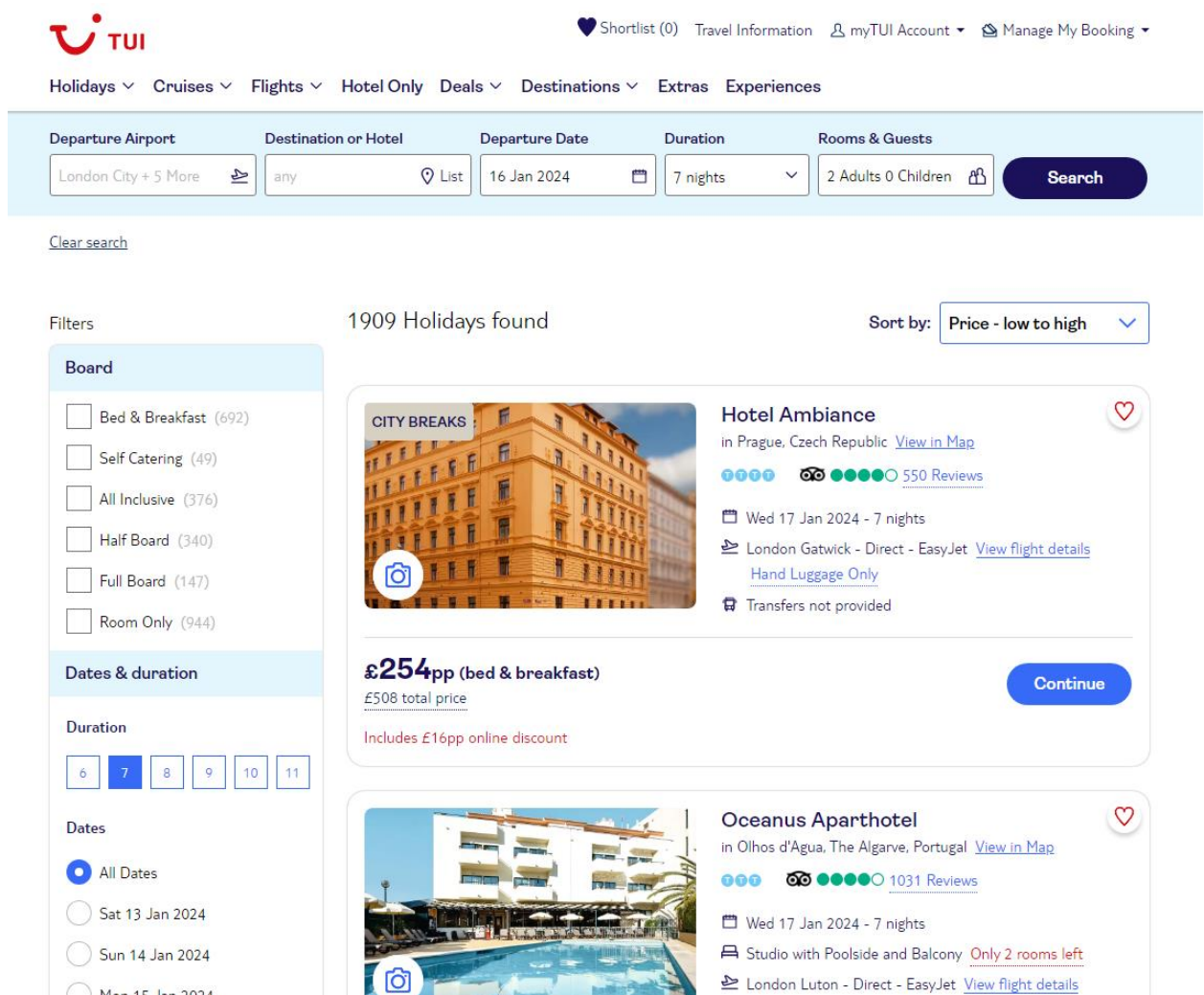
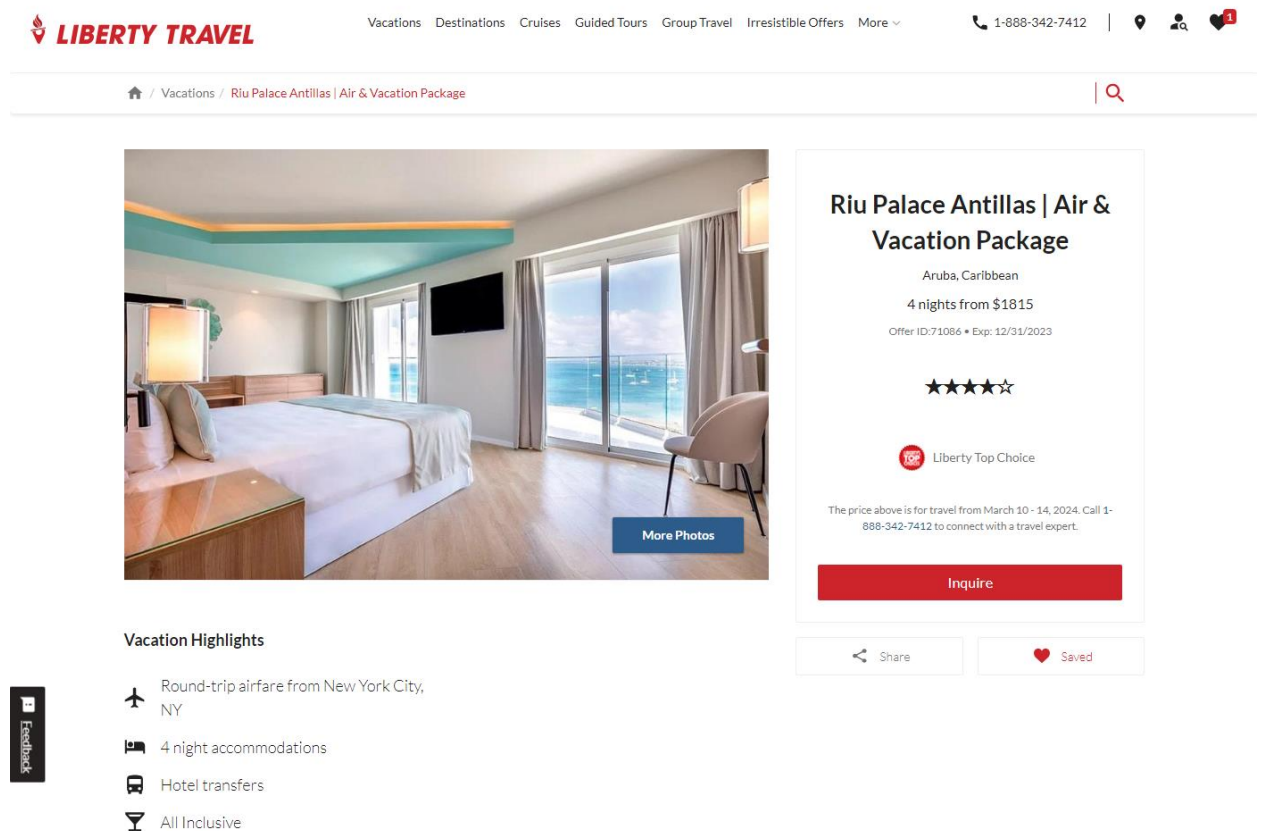


Рисунок 1.1 – Пошук турів на вебсайті туристичної компанії TUI

Форма пошуку дозволяє задати місця відправлення та прибуття, дату початку, тривалість подорожі, кількість гостей, кімнат в якості основних параметрів, а також цілу множину інших, наприклад, включаючи тип туристичного пакета. Окрім варіанта пошуку є цілий ряд варіантів доступу до туристичних пакетів, що можна вважати особливостями програмного забезпечення у цій предметній області: туристичні пакети можуть відображатися за країнами призначення, перелік яких доступний через меню, так само як і набір варіантів подорожей. Користувач може поставити вподобання готелю, але рейтинг визначається самою туристичною компанією або даними з вебсайту Tripadvisor, тобто власної системи відгуків або певних реакцій для самих клієнтів не надається.

Другий аналог, який було виділено, представлений туристичною компанією Liberty Travel (рис. 1.2) [3].



LIBERTY TRAVEL Vacations Destinations Cruises Guided Tours Group Travel Irresistible Offers More ▾ 1-888-342-7412 | 📍 👤 ❤️

🏠 / Vacations / Riu Palace Antillas | Air & Vacation Package 🔍



Riu Palace Antillas | Air & Vacation Package
Aruba, Caribbean
4 nights from \$1815
Offer ID: 71086 • Exp: 12/31/2023
★★★★☆
Liberty Top Choice
The price above is for travel from March 10 - 14, 2024. Call 1-888-342-7412 to connect with a travel expert.
Inquire

Vacation Highlights

- ✈ Round-trip airfare from New York City, NY
- 🏨 4 night accommodations
- 🚗 Hotel transfers
- 🍹 All Inclusive

Feedback

Рисунок 1.2 – Перегляд туристичного пакета на вебсайті туристичної компанії Liberty Travel

Вебсайт Liberty Travel, так само як і попередній, дозволяє створювати власні закладки, повертатися до перегляду таким чином. Загалом функціональність дуже подібна на вебсайт TUI. Присутня можливість створити замовлення туристичного пакету, однак фактично це замовлення з'єднання з консультантом, адже не можна обрати конкретну дату, а тим більше сплатити за замовлення. Фактично задається тільки напрямок подорожі, період подорожі (у місяцях) та залишається інформація клієнтом про себе. У той же час вебсайт компанії TUI дозволяє сформувати повноцінне замовлення.

Вебсайт компанії Viator [4] (рис. 1.3) має дещо відмінну функціональність. Він також як і вебсайт TUI дозволяє виконувати замовлення, але він має і можливість створення оцінок за туристичними пакетами або окремими екскурсіями, адже дана компанія надає широкий спектр послуг, до якого входять і окремі екскурсії.

The screenshot displays the Viator website interface. At the top, there's a search bar with 'New York City' entered. Below the search bar, navigation links include 'Top New York City activities', 'Top New York City landmarks', 'Beyond New York City', and 'New York City trip ideas'. The main content area features a large image of a Circle Line ferry boat on the water with the Statue of Liberty in the background. To the left of the main image is a vertical strip of smaller images. To the right, a booking panel shows the price '\$98.00', the date 'Tuesday, Jan 16, 2024', and the number of people '2'. It includes buttons for 'Reserve Now & Pay Later' and 'Book Now'. Below the booking panel, there's a section for 'Free cancellation' and 'Reserve Now & Pay Later' details. At the bottom, there's a footer with icons for '2 hours (approx.)', 'Mobile ticket', and 'Offered in: English and 6 more'.

Рисунок 1.3 – Перегляд туристичного пакета на вебсайті туристичної компанії Viator

Оцінювання виконується за 5-бальною шкалою і стосується пакетів або екскурсій. Вебсайт не має сторінок окремих міст або країн, тобто фактично і пошук, і окремі сторінки стосуються саме базових послуг компанії, якими є екскурсії і подорожі, виражені туристичними пакетами. Також є можливість створювати власні закладки і повертатися до збережених даних.

1.2 Порівняння розроблюваної програми з існуючими

До детального порівняння було винесено TUI і Viator, адже вони мають ширшу функціональність, та програму, що розробляється (табл. 1.1).

Таблиця 1.1 – Характеристика розроблюваної та існуючих програм

Критерій	TUI	Viator	Розробка
Засоби пошуку	Туристичні пакети, рейси	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі
Засоби збереження	Туристичні пакети	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі, рейси
Замовлення туристичних пакетів	Так	Так	Так
Засоби реагування	Ні	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі
Визначення вартості туристичного пакета на основі прогнозування	Ні	Ні	Так

Розробка, на відміну від вебсайтів TUI і Viator, має засоби пошуку міст, країн, готелів на додачу до туристичних пакетів, засоби збереження міст, країн, готелів, рейсів на додачу до самих туристичних пакетів, засоби реагування на міста, країни, готелі та туристичні пакети, які відсутні у класичних туристичних компаній зазвичай. Окрім того розробка повинна мати засоби визначення вартості туристичного пакета на основі прогнозування.

1.3 Висновки за розділом 1

З тим, щоб досягти мету розробити програмне забезпечення для підтримки діяльності туристичної компанії щодо прийняття обґрунтованих рішень на основі моніторингу основної діяльності компанії і програмного визначення вартості туристичних пакетів потрібно виконати завдання:

- аналіз програм, які підтримують роботу туристичної компанії з клієнтами;
- дослідження методів і моделей визначення вартості туристичного пакета;
- проектування програмного забезпечення підтримки діяльності туристичної компанії;
- реалізація програми підтримки діяльності туристичної компанії.

Завдання аналізу програм, які підтримують роботу туристичної компанії з клієнтами, виконано в розділі 1. Розглянуто вебсайти туристичних компаній TUI, Viator, Liberty Travel, порівняно ці вебзастосунки з розробкою, яку має бути виконано в роботі.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Використання методів прогнозування для розв'язання задачі визначення вартості туристичного пакета

Визначення вартості туристичного пакета фактично є задачею регресії, при чому результат такої регресії, тобто прогнозування несе рекомендаційний характер. Тобто за наявності відповідної інформації про аналогічні ситуації, в яких було визначено вартість інших туристичних пакетів, потрібно фактично визначити логіку таких рішень і використати її для рекомендації щодо прийняття рішення стосовно вартості іншого туристичного пакета. Саме в цьому і полягає задача визначення вартості туристичного пакета, яка за класифікацією задач машинного навчання є задачею регресії.

Для розв'язання цієї задачі має бути накопичено дані стосовно ряду туристичних пакетів, за якими відомими є значення вхідних ознак, які характеризують сам цей пакет і безпосередню подорож, яка цим пакетом забезпечується, та відомою є вартість цього туристичного пакета для однієї особи. Якщо таке накопичення виконано, то задачу можна вважати поставленою. Для розв'язання задачі потрібно створити модель визначення вартості туристичного пакета, яку може бути застосовано для обробки нових туристичних пакетів.

Оскільки дана задача є задачею прогнозування, то визначення вартості туристичного пакета може виконуватися зокрема через наступні моделі:

- лінійна регресія, яка фактично є найпростішою моделлю, що застосовується для розв'язання задач регресії: це алгоритм навчання з учителем, який моделює математичний зв'язок між змінними (вхідними та вихідною) та робить прогнози для безперервних або числових змінних, при цьому множинна лінійна регресія встановлює зв'язок між незалежними змінними (двома або більше) і відповідною залежною змінною, незалежні

змінні можуть бути неперервними або категоріальними, такий тип регресії допомагає передбачити тенденції, визначити майбутні значення та передбачити наслідки змін [5];

– дерева рішень – це тип алгоритму, який включає керуючі оператори для класифікації даних (хоча загалом алгоритм використовується як для класифікації, так і для прогнозування), починається з однієї точки (або вузла), яка потім розгалужується (або розділяється) у двох або більше напрямках, кожна гілка пропонує різні можливі результати, включаючи різноманітні рішення, доки не буде досягнуто кінцевого результату [6];

– екстремальне градієнтне підсилювання для ансамблів дерев рішень: підсилювання градієнта, також відоме як стохастичний градієнт або машина підсилення градієнта, поєднує ідеї градієнтного спуску та посилення ансамблю для створення алгоритму, який зменшує помилки, коли до послідовності додаються нові дерева рішень, мінімізуючи загальні помилки від роботи всієї моделі шляхом ітераційного обчислення градієнта опуклої функції в напрямку мінімуму [7].

Моделі дерева рішень особливо сприйнятливі до недонавчання, оскільки їх простий дизайн сприяє саме простішій інтерпретації, тому можна використовувати навчання на основі ансамблів дерев рішень замість одного дерева рішень, щоб зменшити ризик недостатнього навчання в підсумку, відповідно архітектура єдиного ансамблю дерев рішень поєднує декілька моделей в одну архітектуру, яка використовує прогнозування, отримані від базового дерева рішень для навчання додаткових моделей [7].

На даний момент існують методи, які дозволяють прогнозувати вартість польотів, тобто фактично квитків на літак, які наприклад, були розглянуті в роботах [8]-[9], але проблема в розрізі прогнозування вартості всього туристичного пакету з відповідним розв'язанням її на основі методу, який описано в наступному підрозділі, розглянута саме в цій роботі.

2.2 Метод визначення вартості туристичного пакета

Метод визначення вартості туристичного пакета вимагає спочатку наявності даних, за якими таке визначення вже було виконано.

Для формування вибірки даних для навчання моделі має бути отримано для кожного туристичного пакета наступні характеристики:

- тип туристичного пакета;
- кількість днів, на який було придбано цей туристичний пакет;
- номер місяця, коли здійснюється подорож;
- наявність святкових днів під час подорожі або незадовго до неї;
- основне місто, куди відбувається подорож;
- початкове місто, з якого подорож розпочинається;
- авіалінія, яка реалізує перельоти в подорожі, або інший перевізник для інших варіантів подорожей;
- середня оцінка, яка виставлена готелям, які використовуються під час цієї подорожі;
- вартість туристичного пакету на одну особу.

Тип туристичного пакета є назвою цього типу, тому воно має бути перетворене на категоріальне значення, тобто на наступному етапі має бути виконано кодування типу, тобто визначено всі доступні типи, якими оперує туристична компанія, а потім перетворено на номер типу.

Кількість днів може бути або присутньою у вибірці спочатку, або обчислена через визначення кількості ночей, яку включає проміжок дат, протягом яких триває подорож за туристичним пакетом.

Наявність святкових днів під час подорожі або незадовго до неї запропоновано визначати наступним чином. Потрібно встановити дату початку подорожі та дату її закінчення. Потім визначити, чи є хоча би один з цих днів, які входять в цей проміжок дат, святковим у країні (базово з якої розпочинається подорож), а також перевірено 3 доби до дати початку подорожі та 3 доби після дати закінчення подорожі. Якщо серед цих дат є

хоча би одна святкова, то встановлюється одна категорія за характеристикою наявності святкових днів під час подорожі або незадовго до неї, а якщо жодна дата не є святковою, то встановлюється інша (друга).

Основне місто, куди відбувається подорож, також має бути закодовано через створення числового коду, однак перед цим потрібно спочатку виділити це основне місто або місце. Оскільки подорож може включати декілька міст, то це є важливим. Основним містом запропоновано вважати те, в якому має бути проведено найбільшу кількість ночей під час подорожі. У випадку якщо таких міст декілька, то запропоновано використовувати перше з них.

Початкове місто, з якого подорож розпочинається, також має бути перетворене на категоріальне значення, для чого спочатку визначається перелік усіх міст, з яких розпочинаються подорожі, що розглядаються під час створення моделі.

Перевізник, який виконує перевезення під час реалізації подорожі за туристичним пакетом, також має бути перетворений на категоріальне значення. Якщо перевезення здійснює не один перевізник, то запропоновано або визначити основного перевізника окремо, або використовувати першого з переліку.

Готелі запропоновано характеризувати не кодами, а визначати за оцінками, які були встановлені їм клієнтами. За кожним готелем має бути визначено оцінку, а потім зі всіх оцінок готелів, де має відбуватися проживання за подорожжю за цим туристичним пакетом, має бути обчислено середнє значення.

Якщо вартість туристичного пакету встановлено тільки на декілька осіб, то має бути обчислено значення шляхом ділення вартості такого пакету на кількість осіб, які можуть подорожувати за ним. Це і є цільовим показником моделі.

Після того, як всі ці значення для кожного туристичного пакета були встановлені, а пакети, для яких встановити ці значення неможливо, були

вилучені з остаточної вибірки, має бути реалізовано створення моделі. Модель визначення вартості туристичного пакета за даного методу має бути створена на основі ансамбля дерев рішень за допомогою її подальшого навчання на основі екстремального градієнтного підсилювання. Саме навчання моделі має виконуватися на основі тієї вибірки даних, яку було сформовано описаним вище чином.

Далі дана навчена модель має використовуватися для визначення вартості вже нового туристичного пакета, для якого таке встановлення не було виконано або для якого є потреба в зміні значення. Для цього такий пакет повинен бути охарактеризований та перетворений на відповідний екземпляр вибірки даних, включаючи всі значення окрім вартості цього туристичного пакету. Це перетворення, тобто обчислення фактично значень характеристик має виконуватися у відповідності з процедурою, описаною вище для кожного показника. Після того, як всі такі значення отримано, вони мають бути подані на вхід моделі, а модель повинна обчислити вихідну вартість туристичного пакету, що і є результатом роботи методу.

2.3 Висновки за розділом 2

У цьому розділі виконано визначення задачі визначення вартості туристичного пакета та розглянуто основні методи, які для розв'язання такої задачі можуть використовуватися.

Наукова новизна роботи полягає в запропонованому методі визначення вартості туристичного пакета, який визначає обчислення вхідних ознак моделі на основі характеристик туристичного пакета, виконує створення моделі на основі екстремального градієнтного підсилювання для навчання ансамблів дерев рішень.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Визначення сценаріїв використання програми користувачами

Функціональні вимоги, на основі яких визначені сценарії використання програми:

- відображення інформації про країну, місто, туристичний пакет, готель, кімнату в готелі;
- пошук країн, міст, туристичних пакетів, готелів, кімнат в готелі;
- пошук можливих дат поїздки;
- демонстрування всіх внесених країн, міст, груп туристичних пакетів, готелів, кімнат в готелі, рейсів;
- створення і редагування даних країн, міст, готелів, кімнат у готелі, рейсів;
- демонстрування туристичних пакетів за групою;
- замовлення туристичного пакета для подорожі;
- демонстрування власних замовлень туристичних пакетів для подорожей;
- демонстрування замовлень туристичних пакетів за працівником;
- демонстрування необроблених замовлень туристичних пакетів;
- демонстрування клієнтів за датою поїздки за туристичним пакетом, за готелем, за рейсом;
- підтвердження замовлення туристичного пакета;
- відхилення замовлення туристичного пакета;
- фіксування сплати за замовлення туристичного пакета;
- створення нового туристичного пакета, групи туристичних пакетів;
- демонстрування дат поїздок за туристичним пакетом;
- додавання, змінювання параметрів або скасування дати поїздки за туристичним пакетом;
- демонстрування варіантів вартості туристичного пакета на проміжок часу;

- додавання, змінювання варіантів вартості туристичного пакета;
- змінювання параметрів туристичного пакета, групи туристичних пакетів;
- реагування на країни, міста, готелі, туристичні пакети;
- демонстрування власних реакцій;
- демонстрування реакцій за країною, містом, готелем, туристичним пакетом;
- демонстрування реакцій клієнта за країною, містом, готелем, туристичним пакетом;
- збереження або вилучення зі збережених країни, міста, готелю, туристичного пакету, рейсу до закладок;
- демонстрування країн, міст, готелів, туристичних пакетів, рейсів у власних закладках;
- демонстрування готелів за містом;
- демонстрування найкращих готелів, міст, країн, туристичних пакетів за реакціями;
- авторизація користувачів;
- демонстрування експертних пакетів туристичних подорожей;
- змінювання параметрів облікового запису;
- керування деталізацією туристичного пакету за конкретною датою;
- визначення отриманих сум за туристичним пакетом;
- визначення вартості туристичного пакета на основі прогнозування.

Функціональні вимоги були оброблені і перетворені на аналогічні сценарії або з формуванням додаткових сценаріїв чи об'єднанням їх за необхідності, в залежності від того, як саме передбачається використання програми користувачем (рис. 3.1).

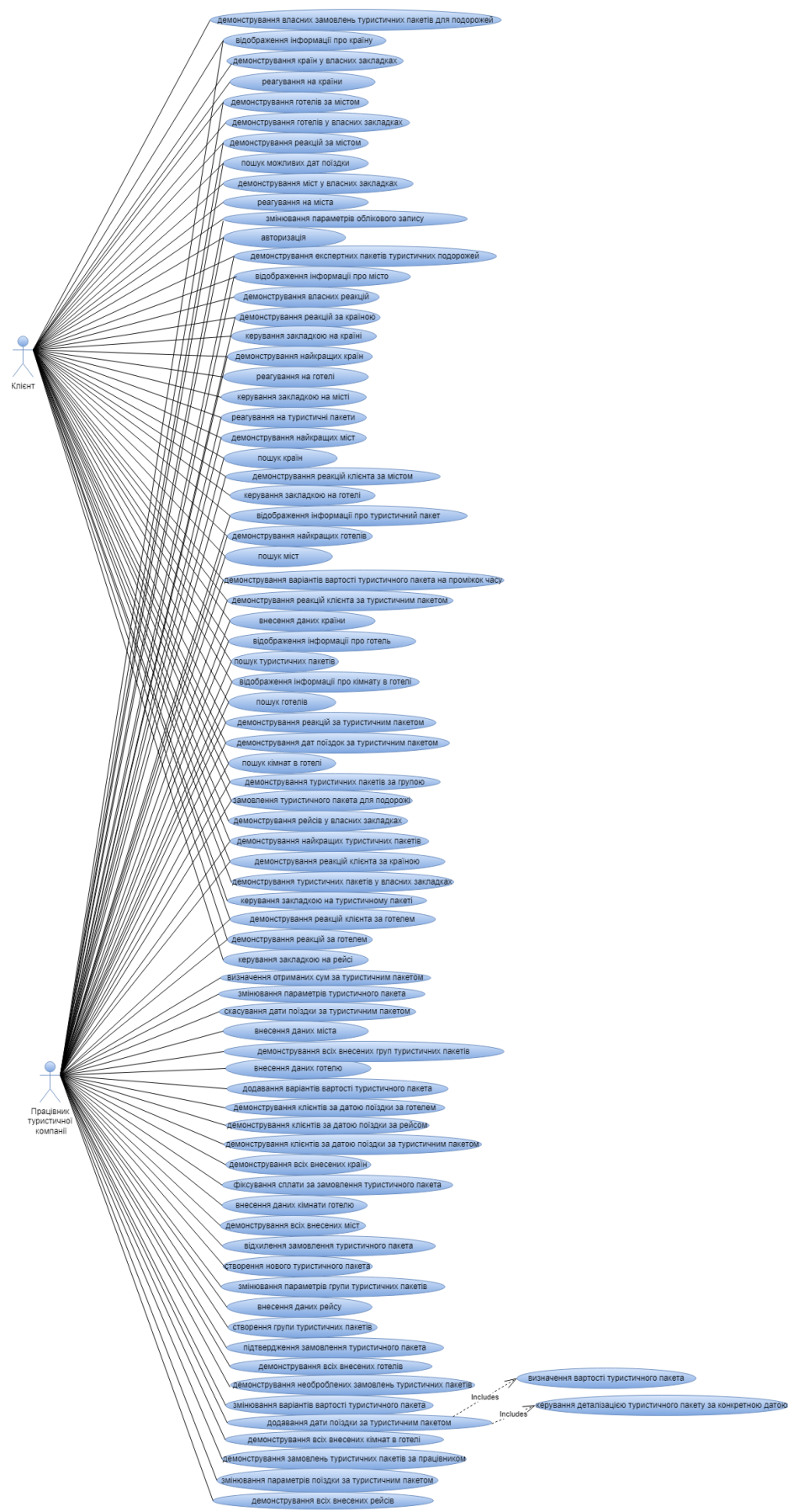


Рисунок 3.1 – Діаграма варіантів використання програми для підтримки діяльності туристичної компанії

3.2 Рішення щодо вибору засобів розробки

Рішення щодо вибору засобів веброзробки передбачало цілісний вибір технологій, що в підсумку зведено до вибору між системою контролю вмісту Wordpress і фреймворком Django [10], які можуть забезпечувати веброзробку (табл. 3.1). Проте Wordpress є найкращим варіантом для типових рішень, а даний вебзастосунок таким не є, адже потребує реалізації методу визначення вартості туристичного пакета, де Wordpress потребує створення плагінів, відповідно написання коду, а готових бібліотек машинного навчання для мови PHP небагато, вони значно поступаються мові Python, до того ж весь процес визначення функцій на рівні коду для Django простіший, адже не всі функціональні вимоги можуть бути забезпечені плагінами Wordpress.

Таблиця 3.1 – Порівняння засобів для розробки програмного забезпечення підтримки діяльності туристичної компанії

Критерій	Wordpress	Django
Призначення	Створення вебзастосунків шляхом підключення готових рішень	Створення вебзастосунків шляхом написання коду
Мова програмування	Не потрібна для створення базових рішень, інакше – PHP	Python
Реалізація методу визначення вартості туристичного пакета	Ускладнена	Значно простіше, багато готових бібліотек
Подальше обслуговування	Постійне	Простіше

3.3 Визначення структури даних у базі даних

Взаємодія різних типів користувачів з програмою, виконання прогнозування на основі методу, представленого в розділі 2, – все це потребує роботи з даними, які тому логічно було структурувати за допомогою бази даних (БД).

Зважаючи на те, що програма орієнтована на конкретну туристичну компанію, а відповідно в її маршрутах використовуються міста і країни, адже саме на поїздки в конкретні міста або країни і направлені пакети подорожей, самі подорожі, то окрім цих сутностей основними сутностями БД є:

- група пакетів туристичних подорожей `Travelpackagegroup`, що дозволяє об'єднувати окремі пакети, що важливо для тих ситуацій, коли пакет має декілька варіантів реалізації, але всі вони мають бути клієнту презентовані разом, щоб він бачив цю можливість вибору;
- пакет туристичної подорожі `Travelpackage`, який вже безпосередньо містить конкретні міста `City`, готелі `Hotel`, рейси `Flight` (переїзди), які входять до цього пакету, і саме він на конкретну дату `Travelpackagedate` продається клієнтам, які купляють ці пакети за вартість, визначену `Travelpackageprice`;
- кімната готелю, де буде проживати клієнт, `Hotelroom`;
- облікові записи користувачів, включаючи працівників туристичної компанії `Employee`, експертів, які є обличчям деяких туристичних пакетів, `Expert`, клієнтів `Client`;
- замовлення туристичного пакета `Application`, яке може оформлюватися на декількох клієнтів `Applicationclient` або шляхом додавання даних про людей, які не є клієнтами в програмі, `Applicationclientout`;
- реакції клієнтів на міста, готелі, країни, пакети, які вони відвідали або за якими вони подорожували, що є багатофункціональними сутностями, адже реакції можуть бути у вигляді уподобання, встановлення конкретної оцінки, а можуть також ще й містити і опис (текст), який фактично передбачається у реакції, але може бути не вказаний клієнтом;

– закладки на міста, країни, готелі, туристичні пакети, рейси, до яких клієнт хоче повернутися і переглянути повторно.

Далі представлені вже безпосередньо всі таблиці БД з деталізацією полів.

Країна фактично є не тільки варіантом вибору, який визначається під час пошуку або як складова у структурі туру, вона представлена також інформацією про себе, що має бути поданою на відповідній сторінці, тому доцільно було не програмно виділяти перелік країн, а саме створити сутність Country з полями:

- id – поле ідентифікатора країни, у яку надає подорожі туристична компанія;
- title – назва країни, якою вона має бути подана в програмі;
- annotation – анотація з описом країни;
- profileimage – профільна картинка, що представляє країну;
- reaction – середня оцінка за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цю країну.

Країна в пошуку задається профільною картинкою, проте для ознайомлення на сторінці самої країни з нею клієнтами можуть надаватися окремі додаткові картинки Countryimage, що має поля:

- id – поле ідентифікатора картинки країни;
- country – країна, яку показано на картинці;
- image – саме зображення країни.

Міста також представлені інформацією про них, а не тільки назвою, тому таблиця City має поля:

- id – поле ідентифікатора міста, в яке надає подорожі туристична компанія;
- country – країна, в якій знаходиться місто;
- title – назва міста, якою воно має бути подано в програмі;
- annotation – анотація з описом міста;
- profileimage – профільна картинка, що представляє місто;

– reaction – середня оцінка за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на це місто.

Аналогічно країнам міста в пошуку представлені профільною картинкою, але додаткові картинки можуть подаватися для міста в таблиці Cityimage з полями:

- id – поле ідентифікатора картинки міста;
- country – місто, яке показано на картинці;
- image – саме зображення міста.

Групи туристичних пакетів знаходяться в таблиці Travelpackagegroup, до якої належать поля:

- id – поле ідентифікатора групи туристичних пакетів;
- title – повна назва, яку дано групі туристичних пакетів;
- slug – текстовий ідентифікатор, що унікально представляє групу туристичних пакетів і використовується для визначення відповідної сторінки в програмі;
- shortname – коротка назва, якою група туристичних пакетів подається, наприклад, у пошуку;
- annotation – анотація з описом групи туристичних пакетів;
- profileimage – профільна картинка, що представляє групу туристичних пакетів.

Туристичний пакет Travelpackage має поля:

- id – поле ідентифікатора туристичного пакета;
- travelpackagegroup – група туристичних пакетів, куди відноситься цей пакет;
- packagetype – тип туристичного пакета;
- title – повна назва, яку дано туристичному пакету;
- slug – текстовий ідентифікатор, що унікально представляє туристичний пакет і використовується для визначення відповідної сторінки в програмі;

- shortname – коротка назва, якою туристичний пакет подається, наприклад, у пошуку;
- annotation – анотація з описом туристичного пакета;
- profileimage – профільна картинка, що представляє туристичний пакет;
- reaction – середня оцінка за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей пакет.

Компанії, які організовують перевезення, тобто авіакомпанії, оператори перевезень залізницею, автобусні оператори, знаходяться в таблиці Flightcompany з полями:

- id – поле ідентифікатора компанії-перевізника;
- title – назва компанії-перевізника;
- webaddress – вебадреса сайту компанії перевізника.

Рейси компаній-перевізників, за допомогою яких можна дістатися з однією точки в іншу і які використовуються в пакетах туристичної компанії, знаходяться в таблиці Flight з полями:

- id – поле ідентифікатора рейсу компанії-перевізника;
- transporttype – тип транспорту, яким виконується це перевезення (літак, потяг, автобус, автомобіль);
- company – компанія-перевізник, яка здійснює цей рейс;
- sourcecity – місто, в якому починається рейс;
- destinationcity – місто, в якому закінчується рейс;
- duration – тривалість рейсу;
- reaction – середня оцінка за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей рейс;
- memo – опис або певна особливість цього рейсу.

Деталі відвідування міста за туристичним пакетом визначаються в таблиці Travelpackagecity з полями:

- id – поле ідентифікатора відвідування міста за туристичним пакетом;

- `travelpackage` – туристичний пакет, в межах якого відбувається відвідування цього міста;
- `city` – місто, відвідування якого відбувається за цим туристичним пакетом;
- `annotation` – анотація з описом особливостей відвідування цього міста за туристичним пакетом;
- `duration` – тривалість відвідування міста за цим туристичним пакетом;
- `startday` – номер дня поїздки за цим туристичним пакетом, в який має розпочатися відвідування цього міста;
- `starttime` – час початку відвідування цього міста;
- `endday` – номер дня поїздки за цим туристичним пакетом, в який має закінчитися відвідування цього міста;
- `endtime` – час закінчення відвідування цього міста.

Переміщення між містами в межах пакету відбувається на основі рейсів та знаходиться в таблиці `Travelpackageflight` з полями:

- `id` – поле ідентифікатора переміщення між містами за туристичним пакетом;
- `travelpackage` – туристичний пакет, в межах якого відбувається це переміщення (переїзд);
- `flight` – рейс, яким здійснюється це переміщення;
- `sourcecity` – місто, з якого починається це переміщення (необхідність винесення цієї характеристики обумовлена тим, що рейс може бути з Запоріжжя у Львів, але для цього пакеті з нього використовується ділянка з Дніпра в Тернопіль);
- `destinationcity` – місто, в якому закінчується це переміщення;
- `duration` – тривалість переміщення за цією ділянкою;
- `annotation` – опис або певна особливість цього переміщення в межах рейсу за туристичним пакетом;
- `registrationstarts` – час, коли починається реєстрація на цей рейс.

Номери готелів, в яких відбувається проживання в межах туристичного пакета, знаходяться в таблиці `Travelpackagehotelroom` з полями:

- `id` – поле ідентифікатора проживання в номері готелю в межах туристичного пакета;
- `travelpackage` – туристичний пакет, до якого відноситься це проживання в номері готелю;
- `packageposition` – позиція в плані проживань туристичного пакета, за якою визначено цей номер готелю, тобто кожен туристичний пакет складається з певної кількості проживань, але деякі позиції з цих проживань можуть бути представлені різними варіантами;
- `packagevariant` – варіант туристичного пакета, до якого відноситься такий варіант проживання;
- `hotelroom` – кімната готелю (фактично тип кімнати готелю), в якій відбувається проживання;
- `annotation` – анотація з описом особливостей проживання в цьому номері готелю саме за цього туристичного пакета;
- `startday` – номер дня поїздки за цим туристичним пакетом, в який має розпочатися проживання в цьому готелі;
- `starttime` – час заїзду в цей готель;
- `endday` – номер дня поїздки за цим туристичним пакетом, в який має закінчитися проживання в цьому готелі;
- `endtime` – час виїзду з цього готелю.

Туристичний пакет в різних варіантах може мати різну вартість до того ж в різні періоди часу, тому такі дані представлено в таблиці `Travelpackageprice` з полями:

- `id` – поле ідентифікатора вартості туристичного пакета;
- `travelpackage` – туристичний пакет, для якого встановлено таку вартість;
- `packagevariant` – варіант пакета, якому відповідає ця вартість;

- annotation – опис особливостей цього варіанта вартості туристичного пакета;
- datefrom – дата продажу туристичних пакетів, з якої ця вартість встановлена;
- dateto – дата продажу туристичних пакетів, до якої ця вартість встановлена;
- price – вартість пакета;
- people – кількість осіб, які здійснюють подорож за цього туристичного пакета.

Оскільки клієнти купляють не загальний туристичний пакет, а пакет на конкретну дату, які вже є фактично конкретними подорожами, то можливі дати турів зібрані в таблиці `Travelpackagedate` з полями:

- id – поле ідентифікатора дати туру за пакетом;
- travelpackage – туристичний пакет, до якого відноситься ця дата;
- starttime – дата і час початку туру;
- endtime – дата і час закінчення туру;
- speciality – особливості реалізації туристичного пакета на цю дату;
- maxplaces – максимальна кількість місць для продажу, доступна за цим туристичним пакетом на цю дату;
- places – поточна доступна кількість місць для продажу, доступна за цим туристичним пакетом на цю дату;
- state – поточний стан реалізації цього туру в цю дату (таким чином визначаються затримки, скасування або фактичне виконання).

При реалізації туру в конкретну дату час початку, закінчення певних позицій туристичного пакету може бути змінений, враховуючи різні особливості. Такі особливості знаходяться в окремих таблицях, що деталізують відповідні таблиці, що визначали таку структуру туристичного пакету.

Деталізація переміщень туристичного пакету за конкретною датою знаходиться в таблиці `Travelpackagedateflight` з полями:

- id – поле ідентифікатора деталізації переміщення в межах туристичного пакету;
- travelpackageflight – переміщення туристичного пакета, якого стосується ця деталізація;
- travelpackagedate – дата туристичного пакета, якої стосується ця деталізація переміщення;
- starttime – дата і час початку переміщення за цієї дати туру;
- endtime – дата і час завершення переміщення за цієї дати туру;
- speciality – особливості переміщення за цієї дати в цьому турі;
- state – стан виконання за цією датою.

Деталізація перебування в цьому місті за цього туристичного пакету за конкретною датою знаходиться в таблиці Travelpackagedatecity з полями:

- id – поле ідентифікатора деталізації перебування в цьому місті в межах туристичного пакету;
- travelpackagecity – перебування в місті за цього туристичного пакету, якого стосується ця деталізація;
- travelpackagedate – дата туристичного пакета, якої стосується ця деталізація перебування в місті;
- starttime – дата і час початку переміщення за цієї дати туру;
- endtime – дата і час завершення переміщення за цієї дати туру;
- speciality – особливості переміщення за цієї дати в цьому турі;
- state – стан виконання за цією датою.

Деталізація проживання в готелі за цього туристичного пакету за конкретною датою знаходиться в таблиці Travelpackagedatehotel з полями:

- id – поле ідентифікатора деталізації проживання в готелі в межах туристичного пакету;
- travelpackagehotelroom – проживання в готелі за цього туристичного пакету, якого стосується ця деталізація;
- travelpackagedate – дата туристичного пакета, якої стосується ця деталізація проживання в готелі;

- starttime – дата і час початку проживання в готелі за цієї дати туру;
- endtime – дата і час завершення проживання в готелі за цієї дати туру;
- speciality – особливості проживання в готелі за цієї дати в цьому турі;
- state – стан виконання за цією датою.

Дані працівників туристичної компанії збережені в таблиці Employee з полями:

- id – поле ідентифікатора працівника туристичної компанії;
- user – обліковий запис цього працівника туристичної компанії (враховано аутентифікацію через Django, тому дані облікового запису не повторені в таблиці, а зроблено зв'язок зі стандартною таблицею Django);
- phone – номер телефону працівника туристичної компанії.

Експерти галузі, які рекламують, привертають увагу до туристичних пакетів, представлені таблицею Expert з полями:

- id – поле ідентифікатора експерта туристичної компанії;
- user – обліковий запис цього експерта туристичної компанії;
- phone – номер телефону експерта туристичної компанії;
- annotation – опис вмінь, біографії цього експерта туристичної компанії;
- imageprofile – фотографія експерта туристичної компанії.

Дані про те, які саме експерти туристичної компанії які туристичні пакети рекламують, знаходяться в таблиці Experttravelpackage з полями:

- id – поле ідентифікатора відношення експерта до туристичного пакета;
- travelpackage – туристичний пакет, який рекламує і здійснює цей експерт;
- expert – експерт, який рекламує і здійснює цей туристичний пакет;
- role – роль експерта (реклама, постійна участь, періодична участь тощо).

Клієнти знаходяться в таблиці Client з полями:

- id – поле ідентифікатора клієнта цієї туристичної компанії;
- user – обліковий запис цього експерта туристичної компанії;
- phone – номер телефону експерта туристичної компанії;
- annotation – опис вмінь, біографії цього експерта туристичної компанії;
- imageprofile – фотографія експерта туристичної компанії;
- passportnumber – номер паспорту або паспортні дані клієнта, якщо він їх надав (інакше дані можуть уточнюватися через інші канали комунікації).

Замовлення туристичного пакета клієнтом у туристичної компанії зберігаються в таблиці Application з полями:

- id – поле ідентифікатора замовлення туристичного пакета клієнтом у туристичної компанії;
- client – клієнт, який замовив цей туристичний пакет;
- travelpackagedate – дата реалізації туристичного пакета, на яку було оформлено замовлення туристичного пакета клієнтом у туристичної компанії;
- employee – працівник, який керує виконанням цього замовлення туристичного пакета клієнтом у туристичної компанії;
- ordertime – дата і час, коли було створено замовлення туристичного пакета клієнтом у туристичної компанії;
- paytime – дата і час, коли було сплачено замовлення туристичного пакета клієнтом у туристичної компанії;
- status – стан замовлення туристичного пакета клієнтом у туристичної компанії;
- comment – коментар клієнта до замовлення туристичного пакета клієнтом у туристичної компанії;
- price – вартість замовлення туристичного пакета клієнтом у туристичної компанії.

Дані клієнтів з замовлення туристичного пакета у туристичної компанії знаходяться в таблиці Applicationclient з полями:

- id – поле ідентифікатора клієнта з замовлення туристичного пакета у туристичної компанії, який буде подорожувати разом з клієнтом, який створив замовлення;
- application – замовлення туристичного пакета у туристичної компанії, до якого відноситься цей клієнт;
- client – клієнт, який буде подорожувати разом з клієнтом, який створив замовлення;
- confirmed – підтвердження участі в подорожі зі сторони самого цього клієнта.

Оскільки не всі подорожуючі можуть бути користувачами програми, то дані про деяких користувачів можуть вноситися окремо від облікових записів, які вони можуть не мати. Такі дані щодо участі таких людей у подорожі вносяться в таблицю Applicationclientout з полями:

- id – поле ідентифікатора клієнта без облікового запису з замовлення туристичного пакета у туристичної компанії, який буде подорожувати разом з клієнтом, який створив замовлення;
- application – замовлення туристичного пакета у туристичної компанії, до якого відноситься цей клієнт;
- clientsurname – прізвище клієнта;
- clientname – ім'я клієнта;
- passportnumber – номер паспорту або паспортні дані клієнта, якщо він їх надав.

Готелі, які використовуються туристичною компанією для проживання клієнтів, знаходяться в таблиці Hotel з полями:

- id – поле ідентифікатора готелю;
- title – назва готелю;
- city – місто, де цей готель знаходиться;
- star – кількість зірок, яка присвоєна готелю, якщо такі дані відомі;

- hoteltype – тип готелю;
- pool – наявність басейну в готелі;
- webaddress – вебадреса сайту готелю;
- postaddress – поштова адреса готелю;
- slug – текстовий ідентифікатор, що унікально представляє готель;
- annotation – анотація з описом готелю;
- profileimage – профільна картинка, що представляє готель;
- reaction – середня оцінка за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей готель.

Номери, доступні в готелі, знаходяться в таблиці Hotelroom з полями:

- id – поле ідентифікатора такого номеру (фактично типу номеру) в готелі;
- hotel – готель, в якому знаходиться кімната;
- title – назва такого номеру;
- roomtype – тип номеру, який характеризується цим записом;
- annotation – анотація з описом особливостей такого номеру готелю;
- bath – тип ванної кімнати в такому номері готелю;
- places – кількість місць у такому номері готелю;
- bednumber – кількість спальних місць у такому номері готелю;
- bedtype – тип спальних місць у такому номері готелю;
- bar – наявність бару в такому номері готелю;
- tv – наявність телебачення в такому номері готелю;
- internet – наявність інтернету в такому номері готелю;
- viewtype – вид, що відкривається в такому номері готелю (на море, на сад, відсутній тощо);
- balcony – наявність балкону в такому номері готелю.

Фотографії готельного номеру знаходяться в таблиці Hotelroomimage з полями:

- id – поле ідентифікатора фотографії готельного номеру;
- hotelroom – готельний номер, який показаний на фотографії;

– image – фотографія, яка показує цей готельний номер.

Фотографії самого готелю знаходяться в таблиці Hotelimage з полями:

- id – поле ідентифікатора фотографії готелю;
- hotel – готель, який показаний на фотографії;
- image – фотографія, яка показує цей готель.

Реакції клієнтів на міста, які вони відвідували, знаходяться в таблиці Cityreaction з полями:

- id – поле ідентифікатора реакції клієнта на місто;
- city – місто, до якого адресована реакція;
- client – клієнт, який створив цю реакцію;
- reactiontype – тип реакції (бінарна – уподобання, шкала – 10 значень);
- reactionvalue – значення реакції (уподобання або ні, або значення за 10-бальною шкалою в залежності від типу реакції);
- reactionview – текстова інформація стосовно реакції;
- reactiontime – час збереження реакції.

Реакції клієнтів на готелі, в яких вони проживали, знаходяться в таблиці Hotelreaction з полями:

- id – поле ідентифікатора реакції клієнта на готель;
- hotel – готель, до якого адресована ця реакція;
- client – клієнт, який створив цю реакцію;
- reactiontype – тип реакції (бінарна – уподобання, шкала – 10 значень);
- reactionvalue – значення реакції (уподобання або ні, або значення за 10-бальною шкалою в залежності від типу реакції);
- reactionview – текстова інформація стосовно реакції;
- reactiontime – час збереження реакції.

Закладки клієнтами міст для повернення в подальшому знаходяться в таблиці Citybookmark з полями:

- id – ідентифікатор закладки клієнтом міста;

- city – місто, за сторінкою якого створено закладку;
- client – клієнт, який створив закладку.

Закладки клієнтами країн для повернення в подальшому знаходяться в таблиці Countrybookmark з полями:

- id – ідентифікатор закладки клієнтом країни;
- country – країна, за сторінкою якої було створено закладку;
- client – клієнт, який створив закладку.

Закладки клієнтами готелів для повернення в подальшому знаходяться в таблиці Hotelbookmark з полями:

- id – ідентифікатор закладки клієнтом готелю;
- hotel – готель, за сторінкою якого було створено закладку;
- client – клієнт, який створив закладку.

Закладки клієнтами туристичних пакетів для повернення в подальшому знаходяться в таблиці Travelpackagebookmark з полями:

- id – ідентифікатор закладки клієнтом туристичного пакету;
- travelpackage – туристичний пакет, за сторінкою якого було створено закладку;
- client – клієнт, який створив закладку.

Закладки клієнтами туристичних рейсів для повернення в подальшому знаходяться в таблиці Flightbookmark з полями:

- id – ідентифікатор закладки клієнтом рейсу;
- flight – рейс для переміщення з одного міста в інше, за сторінкою якого було створено закладку;
- client – клієнт, який створив закладку (рис. 3.2).

Для підтримки БД обрано систему PostgreSQL, достатньо потужну і безкоштовну.

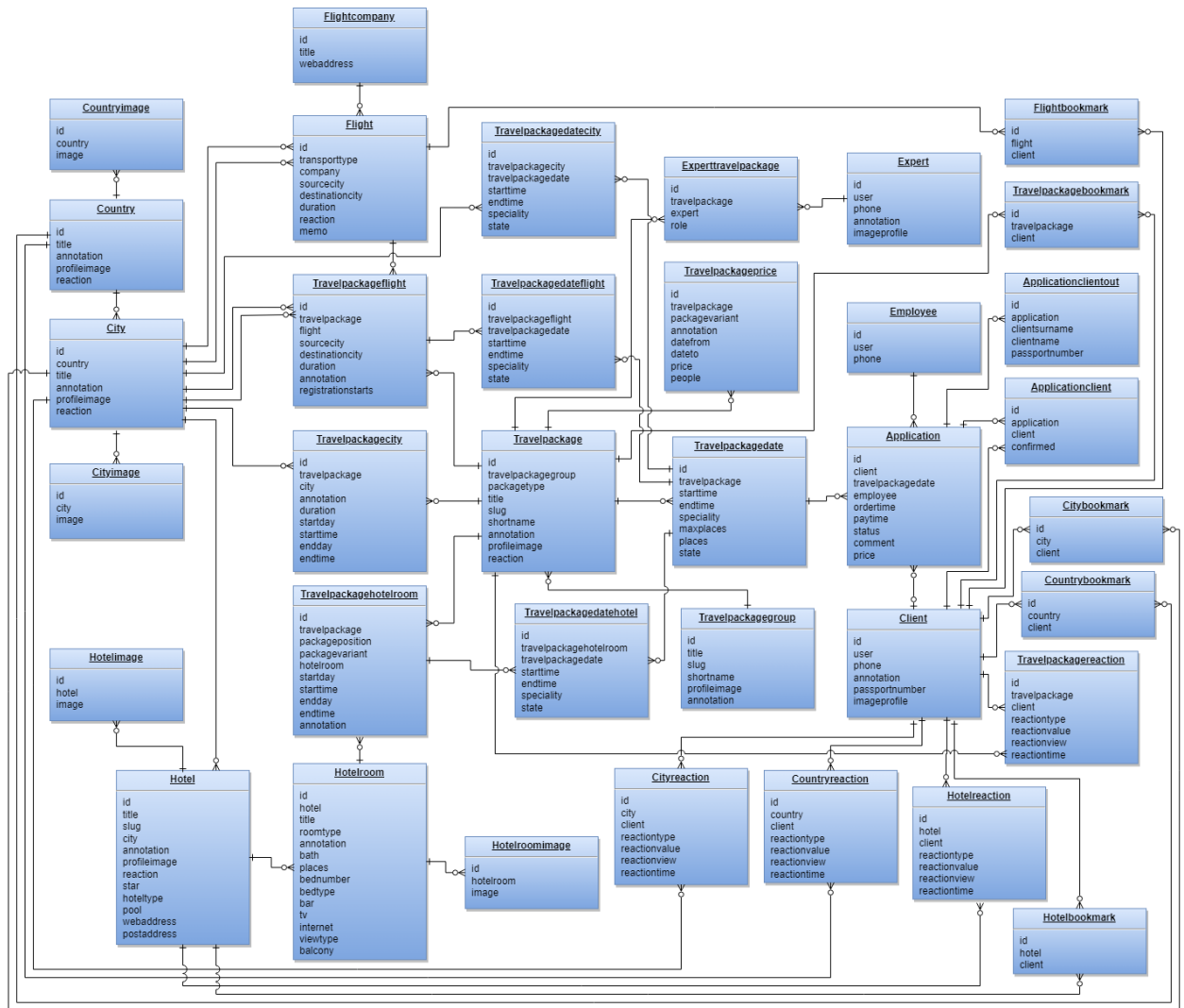


Рисунок 3.2 – Схема БД

Реакції клієнтів на країни, які вони відвідували, знаходяться в таблиці Countryreaction з полями:

- id – поле ідентифікатора реакції клієнта на країну;
- country – країна, до якого адресована ця реакція;
- client – клієнт, який створив цю реакцію;
- reactiontype – тип реакції (бінарна – уподобання, шкала – 10 значень);
- reactionvalue – значення реакції (уподобання або ні, або значення за 10-бальною шкалою в залежності від типу реакції);
- reactionview – текстова інформація стосовно реакції;
- reactiontime – час збереження реакції.

Реакції клієнтів на туристичні пакети, за якими вони подорожували, знаходяться в таблиці `Travelpackagereaction` з полями:

- `id` – поле ідентифікатора реакції клієнта на туристичний пакет;
- `travelpackage` – туристичний пакет, до якого адресована ця реакція;
- `client` – клієнт, який створив цю реакцію;
- `reactiontype` – тип реакції (бінарна – уподобання, шкала – 10 значень);
- `reactionvalue` – значення реакції (уподобання або ні, або значення за 10-бальною шкалою в залежності від типу реакції);
- `reactionview` – текстова інформація стосовно реакції;
- `reactiontime` – час збереження реакції.

3.4 Висновки за розділом 3

Проектування програми виконано повністю, визначивши сценарії роботи з програмою, обравши для розробки використання фреймворку Django, що визначає архітектуру застосунку, який має бути розроблено далі, визначивши структуру БД.

4 РЕАЛІЗАЦІЯ ПРОГРАМИ

4.1 Визначення послідовності роботи з програмою

Визначення послідовності роботи користувача з програмою було виконано на основі розділення доступних функцій на дві групи (рис. 4.1).

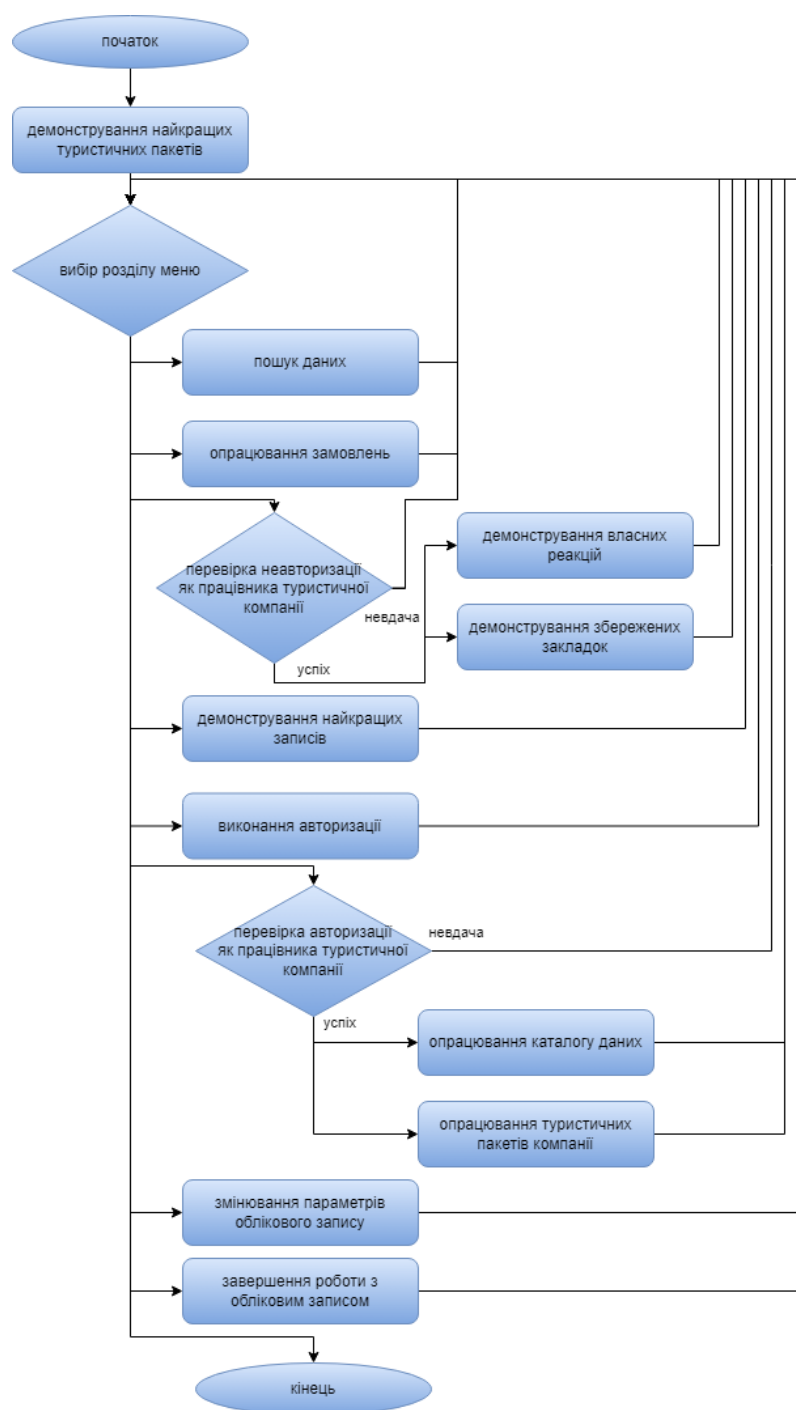


Рисунок 4.1 – Функціональна схема програми підтримки діяльності туристичної компанії

Перша з груп функцій програми доступна користувачам після авторизації як клієнтів та до авторизації, а друга – тільки працівникам туристичної компанії, які попередньо були авторизовані. Частина функцій першої групи потребує авторизації, але вони доступні всім користувачам, проте вже при зверненні відбудеться повідомлення про неможливість такого доступу. Частина функцій першої групи недоступна для працівника туристичної компанії.

4.2 Особливості реалізації програми

Програма створена за принципом шаблону проєктування Model View Template (MVT), що визначений безпосередньо самою структурою Django-застосунку.

Функція `countrypageview` реалізує представлення відображення інформації про країну, яка визначається своєю назвою на основі параметра функції `countrytitle`. Окрім безпосередньо даних країни на сторінку, що представлена шаблоном `country.html`, створеним на базі шаблону [11], передаються картинки цієї країни, 10 міст з найвищим рейтингом за реакціями клієнтів, 10 найкращих туристичних пакетів, які включають до себе міста цієї країни. Також для авторизованого користувача на основі його реакцій та закладок визначається, чи ця інформація була збережена клієнтом, чи ні. Аналогічно передачу інформації про місто, туристичний пакет, готель, кімнату в готелі реалізовано функціями відповідно `citypageview`, `travelpackagepageview`, `hotelpageview`, `hotelroompageview`.

Демонстрування всіх внесених даних створено функціями представлень для країн – функцією `employeecountriespageview`, міст – функцією `employeecitiespageview`, груп туристичних пакетів – функцією `employeetravelpackagegroupspageview`, туристичних пакетів – функцією `employeetravelpackagespageview`, готелів – функцією `employeehotelspageview`,

кімнат в готелі – функцією `employeehotelroomspageview`, рейсів – функцією `employeeflightspageview`.

Демонстрування туристичних пакетів за групою реалізовано функцією представлення `grouptravelpackagesview`, яка має параметр `travelpackagegroupslug`, що є адресою (текстовим ідентифікатором) групи туристичних пакетів. Так само окрім самих туристичних пакетів шаблон сторінки заповнюється реакціями та закладками стосовно цих пакетів для авторизованого клієнта, якщо користувач авторизувався і він є клієнтом.

Функція `cityhotelspageview` реалізує представлення демонстрування готелів за містом, заповнюючи шаблон вебсторінки `cityhotels.html` даними готелів за рейтингом.

Функція `besthotelspageview` реалізує представлення демонстрування найкращих готелів, `bestcountriespageview` – відповідно найкращих країн, `bestcitiespageview` – найкращих міст, `besttravelpackagespageview` – найкращих туристичних пакетів.

Функція `expertpackagespageview` реалізує представлення демонстрування експертних пакетів туристичних подорожей, обираючи їх для шаблону вебсторінки за логіном експерта `expertname`.

Демонстрування країн, міст, готелів, туристичних пакетів, рейсів у власних закладках реалізовано функціями представлення `bookmarkedcountriespageview`, представлення `bookmarkedcitiespageview`, `bookmarkedhotelspageview`, представлення `bookmarkedpackagespageview`, представлення `bookmarkedflightspageview` відповідно.

Демонстрування реакцій клієнта за країною, містом, готелем, туристичним пакетом реалізовано функцією представлення `clientreactionspageview`, що наповнює шаблон вебсторінки `clientreactions.html` даними через словник з ключами, що відповідають переліченим варіантам реакцій: `countries`, `cities`, `packages`, `hotels`. Логін користувача передається через параметр `clientname`, а якщо це значення пропущене, то використовуються дані авторизації, що таким чином реалізує перегляд власних реакцій.

Демонстрування реакцій за країною, містом, готелем, туристичним пакетом реалізовано функціями представлення `countryreactionspageview`, де параметр `countrytitle` визначає назву країни, представлення `cityreactionspageview`, де параметр `citytitle` визначає назву міста, представлення `hotelreactionspageview`, де параметр `hotelslug` визначає текстовий ідентифікатор готелю, представлення `packagereactionspageview`, де параметр `packageslug` визначає текстовий ідентифікатор туристичного пакету.

Збереження країни, міста, готелю, туристичного пакету, рейсу до закладок виконано функцією представлення `countrybookmarkview`, у параметрі його `countrytitle` задається назва країни, а дані облікового запису визначаються за авторизацією, та аналогічними функціями `citybookmarkview`, `hotelbookmarkview`, `packagebookmarkview`, `flightbookmarkview` відповідно. У функціях також перевіряється, чи створено такі закладки вже, а якщо створено, то виконується вилучення такої закладки.

Визначення отриманих сум за туристичним пакетом реалізовано функцією представлення `travelpackageoutcomeview`, що має параметр `travelpackageslug`, що визначає унікальний текстовий ідентифікатор туристичного пакету, та параметри `startdate` і `enddate`, що визначають відповідно період визначення. Аналогічно були створені інші функції представлень.

Модель країн, у які надає подорожі туристична компанія, реалізована класом `Country`, для реалізації якого забезпечено створення полів:

- поле `title`, яке має тип `CharField` і визначає назву країни, якою вона має бути подана в програмі;
- поле `annotation`, яке має тип `TextField` і визначає анотацію з описом країни;
- поле `profileimage`, яке має тип `ImageField` і визначає профільну картинку, що представляє країну;

– поле `reaction`, яке має тип `FloatField` і визначає середню оцінку за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цю країну.

Модель додаткової картинки країни реалізована класом `Countryimage`, для реалізації якого забезпечено створення полів:

– поле `country`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає країну, яку показано на картинці;

– поле `ImageField`, яке має тип `CharField` і визначає саме зображення країни.

Модель міст, у які надає подорожі туристична компанія, реалізована класом `City`, для реалізації якого забезпечено створення полів:

– поле `country`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає країну, в якій знаходиться місто;

– поле `title`, яке має тип `CharField` і визначає назву міста, якою воно має бути подано в програмі;

– поле `annotation`, яке має тип `TextField` і визначає анотацію з описом міста;

– поле `profileimage`, яке має тип `ImageField` і визначає профільну картинку, що представляє місто;

– поле `reaction`, яке має тип `FloatField` і визначає середню оцінку за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на це місто.

Модель додаткової картинки міста реалізована класом `Cityimage`, для реалізації якого забезпечено створення полів:

– поле `country`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, яке показано на картинці;

– поле `image`, яке має тип `ImageField` і визначає саме зображення міста.

Модель групи туристичних пакетів реалізована класом `Travelpackagegroup`, для реалізації якого забезпечено створення полів:

- поле `title`, яке має тип `CharField` і визначає повну назву, яку дано групі туристичних пакетів;
- поле `slug`, яке має тип `SlugField` і визначає текстовий ідентифікатор, що унікально представляє групу туристичних пакетів і використовується для визначення відповідної сторінки в програмі;
- поле `shortname`, яке має тип `CharField` і визначає коротку назву, якою група туристичних пакетів подається, наприклад, у пошуку;
- поле `annotation`, яке має тип `TextField` і визначає анотацію з описом групи туристичних пакетів;
- поле `profileimage`, яке має тип `ImageField` і визначає профільну картинку, що представляє групу туристичних пакетів.

Модель туристичного пакета реалізована класом `Travelpackage`, для реалізації якого забезпечено створення полів:

- поле `travelpackagegroup`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає групу туристичних пакетів, куди відноситься цей пакет;
- поле `packagetype`, яке має тип `CharField` і визначає тип туристичного пакета як вибір з переліку;
- поле `title`, яке має тип `CharField` і визначає повну назву, яку дано туристичному пакету;
- поле `slug`, яке має тип `SlugField` і визначає текстовий ідентифікатор, що унікально представляє туристичний пакет і використовується для визначення відповідної сторінки в програмі;
- поле `shortname`, яке має тип `CharField` і визначає коротку назву, якою туристичний пакет подається, наприклад, у пошуку;
- поле `annotation`, яке має тип `TextField` і визначає анотацію з описом туристичного пакета;
- поле `profileimage`, яке має тип `ImageField` і визначає профільну картинку, що представляє туристичний пакет;

- поле `reaction`, яке має тип `FloatField` і визначає середню оцінку за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей пакет.

Модель компанії-перевізника реалізована класом `Flightcompany`, для реалізації якого забезпечено створення полів:

- поле `title`, яке має тип `CharField` і визначає назву компанії-перевізника;
- поле `webaddress`, яке має тип `URLField` і визначає веб адресу сайту компанії перевізника.

Модель рейсу компанії-перевізника реалізована класом `Flight`, для реалізації якого забезпечено створення полів:

- поле `transporttype`, яке має тип `CharField` і визначає тип транспорту, яким виконується це перевезення (літак, потяг, автобус, автомобіль);
- поле `company`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає компанію-перевізника, яка здійснює цей рейс;
- поле `sourcecity`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, в якому починається рейс;
- поле `destinationcity`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, в якому закінчується рейс;
- поле `duration`, яке має тип `TimeField` і визначає тривалість рейсу;
- поле `reaction`, яке має тип `FloatField` і визначає середню оцінку за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей рейс;
- поле `memo`, яке має тип `CharField` і визначає опис або певну особливість цього рейсу.

Модель деталізації відвідування міста за туристичним пакетом реалізована класом `Travelpackagecity`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, в межах якого відбувається відвідування цього міста;
- поле `city`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, відвідування якого відбувається за цим туристичним пакетом;
- поле `annotation`, яке має тип `CharField` і визначає анотацію з описом особливостей відвідування цього міста за туристичним пакетом;
- поле `duration`, яке має тип `TimeField` і визначає тривалість відвідування міста за цим туристичним пакетом;
- поле `startday`, яке має тип `PositiveSmallIntegerField` і визначає номер дня поїздки за цим туристичним пакетом, в який має розпочатися відвідування цього міста;
- поле `starttime`, яке має тип `TimeField` і визначає час початку відвідування цього міста;
- поле `endday`, яке має тип `PositiveSmallIntegerField` і визначає номер дня поїздки за цим туристичним пакетом, в який має закінчитися відвідування цього міста;
- поле `endtime`, яке має тип `TimeField` і визначає час закінчення відвідування цього міста.

Модель переміщення між містами в межах пакету реалізована класом `Travelpackageflight`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, в межах якого відбувається це переміщення (переїзд);
- поле `flight`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає рейс, яким здійснюється це переміщення;
- поле `sourcecity`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, з якого починається це переміщення;

- поле `destinationcity`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, в якому закінчується це переміщення;
- поле `duration`, яке має тип `TimeField` і визначає тривалість переміщення за цією ділянкою;
- поле `annotation`, яке має тип `CharField` і визначає опис або певну особливість цього переміщення в межах рейсу за туристичним пакетом;
- поле `registrationstarts`, яке має тип `TimeField` і визначає час, коли починається реєстрація на цей рейс.

Модель номерів готелів реалізована класом `Travelpackagehotelroom`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, до якого відноситься це проживання в номері готелю;
- поле `packageposition`, яке має тип `PositiveSmallIntegerField` і визначає позицію в плані проживань туристичного пакета, за якою визначено цей номер готелю, тобто кожен туристичний пакет складається з певної кількості проживань, але деякі позиції з цих проживань можуть бути представлені різними варіантами;
- поле `packagevariant`, яке має тип `PositiveSmallIntegerField` і визначає варіант туристичного пакета, до якого відноситься такий варіант проживання;
- поле `hotelroom`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає кімнату готелю;
- поле `annotation`, яке має тип `CharField` і визначає анотацію з описом особливостей проживання в цьому номері готелю саме за цього туристичного пакета;
- поле `startday`, яке має тип `PositiveSmallIntegerField` і визначає номер дня поїздки за цим туристичним пакетом, в який має розпочатися проживання в цьому готелі;

- поле `starttime`, яке має тип `TimeField` і визначає час заїзду в цей готель;
- поле `endday`, яке має тип `PositiveSmallIntegerField` і визначає номер дня поїздки за цим туристичним пакетом, в який має закінчитися проживання в цьому готелі;
- поле `endtime`, яке має тип `TimeField` і визначає час виїзду з цього готелю.

Модель вартості туристичного пакета реалізована класом `Travelpackageprice`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, для якого встановлено таку вартість;
- поле `packagevariant`, яке має тип `PositiveIntegerField` і визначає варіант пакета, якому відповідає ця вартість;
- поле `annotation`, яке має тип `CharField` і визначає опис особливостей цього варіанта вартості туристичного пакета;
- поле `datefrom`, яке має тип `DateField` і визначає дату продажу туристичних пакетів, з якої ця вартість встановлена;
- поле `dateto`, яке має тип `DateField` і визначає дату продажу туристичних пакетів, до якої ця вартість встановлена;
- поле `price`, яке має тип `DecimalField` і визначає вартість пакета;
- поле `people`, яке має тип `PositiveSmallIntegerField` і визначає кількість осіб, які здійснюють подорож за цього туристичного пакета.

Модель дати поїздки за туристичним пакетом реалізована класом `Travelpackagedate`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, до якого відноситься ця дата;
- поле `starttime`, яке має тип `DateTimeField` і визначає дату і час початку туру;

- поле `endtime`, яке має тип `DateTimeField` і визначає дату і час закінчення туру;
- поле `speciality`, яке має тип `CharField` і визначає особливості реалізації туристичного пакета на цю дату;
- поле `maxplaces`, яке має тип `CharField` і визначає максимальна кількість місць для продажу, доступна за цим туристичним пакетом на цю дату;
- поле `places`, яке має тип `PositiveSmallIntegerField` і визначає поточну доступну кількість місць для продажу, доступну за цим туристичним пакетом на цю дату;
- поле `state`, яке має тип `CharField` і визначає поточний стан реалізації цього туру в цю дату.

Модель деталізації переміщень туристичного пакету за конкретною датою реалізована класом `Travelpackageflight`, для реалізації якого забезпечено створення полів:

- поле `travelpackageflight`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає переміщення туристичного пакета, якого стосується ця деталізація;
- поле `travelpackageflight`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає дату туристичного пакета, якої стосується ця деталізація переміщення;
- поле `starttime`, яке має тип `DateTimeField` і визначає дату і час початку переміщення за цієї дати туру;
- поле `endtime`, яке має тип `DateTimeField` і визначає дату і час завершення переміщення за цієї дати туру;
- поле `speciality`, яке має тип `CharField` і визначає особливості переміщення за цієї дати в цьому турі.

Модель деталізації перебування в місті за туристичного пакету реалізована класом `Travelpackagecity`, для реалізації якого забезпечено створення полів:

- поле `travelpackagecity`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає перебування в місті за цього туристичного пакету, якого стосується ця деталізація;
- поле `travelpackagedate`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає дату туристичного пакета, якої стосується ця деталізація перебування в місті;
- поле `starttime`, яке має тип `DateTimeField` і визначає дату і час початку переміщення за цієї дати туру;
- поле `endtime`, яке має тип `DateTimeField` і визначає дату і час завершення переміщення за цієї дати туру;
- поле `speciality`, яке має тип `CharField` і визначає особливості переміщення за цієї дати в цьому турі.

Модель деталізації проживання в готелі за туристичного пакету реалізована класом `Travelpackagedatehotel`, для реалізації якого забезпечено створення полів:

- поле `travelpackagehotelroom`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає проживання в готелі за цього туристичного пакету, якого стосується ця деталізація;
- поле `travelpackagedate`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає дату туристичного пакета, якої стосується ця деталізація проживання в готелі;
- поле `starttime`, яке має тип `DateTimeField` і визначає дату і час початку проживання в готелі за цієї дати туру;
- поле `endtime`, яке має тип `DateTimeField` і визначає дату і час завершення проживання в готелі за цієї дати туру;
- поле `speciality`, яке має тип `CharField` і визначає особливості проживання в готелі за цієї дати в цьому турі.

Модель працівника туристичної компанії реалізована класом `Employee`, для реалізації якого забезпечено створення полів:

- поле `user`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає обліковий запис цього працівника туристичної компанії;
- поле `phone`, яке має тип `CharField` і визначає номер телефону працівника туристичної компанії.

Модель експерта туристичної компанії реалізована класом `Expert`, для реалізації якого забезпечено створення полів:

- поле `user`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає обліковий запис цього експерта туристичної компанії;
- поле `phone`, яке має тип `CharField` і визначає номер телефону експерта туристичної компанії;
- поле `annotation`, яке має тип `TextField` і визначає опис вмінь, біографії цього експерта туристичної компанії;
- поле `imageprofile`, яке має тип `ImageField` і визначає фотографію експерта туристичної компанії.

Модель ролі експерта туристичної компанії в туристичному пакеті реалізована класом `Experttravelpackage`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, який рекламує і здійснює цей експерт;
- поле `expert`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає експерта, який рекламує і здійснює цей туристичний пакет;
- поле `role`, яке має тип `CharField` і визначає роль експерта.

Модель клієнта туристичної компанії реалізована класом `Client`, для реалізації якого забезпечено створення полів:

- поле `user`, яке має тип `ForeignKey` для встановлення зовнішнього ключа з моделі `models.User` підпаketу авторизації `django.contrib.auth` і визначає обліковий запис цього експерта туристичної компанії;
- поле `phone`, яке має тип `CharField` і визначає номер телефону експерта туристичної компанії;

- поле `annotation`, яке має тип `TextField` і визначає опис вмінь, біографії цього експерта туристичної компанії;
- поле `imageprofile`, яке має тип `ImageField` і визначає фотографію експерта туристичної компанії;
- поле `passportnumber`, яке має тип `CharField` і визначає номер паспорту або паспортні дані клієнта, якщо він їх надав.

Модель замовлення туристичного пакета клієнтом у туристичної компанії реалізована класом `Application`, для реалізації якого забезпечено створення полів:

- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який замовив цей туристичний пакет;
- поле `travelpackagedate`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає дату реалізації туристичного пакета, на яку було оформлено замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `employee`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає працівника, який керує виконанням цього замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `ordertime`, яке має тип `DateTimeField` і визначає дату і час, коли було створено замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `paytime`, яке має тип `DateTimeField` і визначає дату і час, коли було сплачено замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `status`, яке має тип `CharField` і визначає стан замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `comment`, яке має тип `CharField` і визначає коментар клієнта до замовлення туристичного пакета клієнтом у туристичної компанії;
- поле `price`, яке має тип `DecimalField` і визначає вартість замовлення туристичного пакета клієнтом у туристичної компанії.

Модель даних клієнтів з замовлення туристичного пакета у туристичної компанії реалізована класом `Applicationclient`, для реалізації якого забезпечено створення полів:

- поле `application`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає замовлення туристичного пакета у туристичної компанії, до якого відноситься цей клієнт;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який буде подорожувати разом з клієнтом, який створив замовлення;
- поле `confirmed`, яке має тип `BooleanField` і визначає підтвердження участі в подорожі зі сторони самого цього клієнта.

Модель даних незареєстрованих клієнтів з замовлення туристичного пакета у туристичної компанії реалізована класом `Applicationclientout`, для реалізації якого забезпечено створення полів:

- поле `application`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає замовлення туристичного пакета у туристичної компанії, до якого відноситься цей клієнт;
- поле `clientsurname`, яке має тип `CharField` і визначає прізвище клієнта;
- поле `clientname`, яке має тип `CharField` і визначає ім'я клієнта;
- поле `passportnumber`, яке має тип `CharField` і визначає номер паспорту або паспортні дані клієнта, якщо він їх надав.

Модель готеля реалізована класом `Hotel`, для реалізації якого забезпечено створення полів:

- поле `title`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає назву готелю;
- поле `city`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, де цей готель знаходиться;
- поле `star`, яке має тип `SmallIntegerField` і визначає кількість зірок, яка присвоєна готелю, якщо такі дані відомі;

- поле `hoteltype`, яке має тип `CharField` і визначає тип готелю;
- поле `pool`, яке має тип `BooleanField` і визначає наявність басейну в готелі;
- поле `webaddress`, яке має тип `URLField` і визначає вебадресу сайту готелю;
- поле `postaddress`, яке має тип `CharField` і визначає поштову адресу готелю;
- поле `slug`, яке має тип `SlugField` і визначає текстовий ідентифікатор, що унікально представляє готель;
- поле `annotation`, яке має тип `TextField` і визначає анотація з описом готелю;
- поле `profileimage`, яке має тип `ImageField` і визначає профільна картинка, що представляє готель;
- поле `reaction`, яке має тип `FloatField` і визначає середню оцінку за 10-бальною шкалою, яку було в реакціях зазначено клієнтами туристичної компанії на цей готель.

Модель номерів, доступних у готелі, реалізована класом `Hotelroom`, для реалізації якого забезпечено створення полів:

- поле `title`, яке має тип `CharField` і визначає назву такого номеру;
- поле `hotel`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає готель, в якому знаходиться цей номер;
- поле `roomtype`, яке має тип `CharField` і визначає тип номеру, який характеризується цим записом;
- поле `annotation`, яке має тип `CharField` і визначає анотацію з описом особливостей такого номеру готелю;
- поле `bath`, яке має тип `CharField` і визначає тип ванної кімнати в такому номері готелю;
- поле `places`, яке має тип `SmallIntegerField` і визначає кількість місць у такому номері готелю;

- поле `bednumber`, яке має тип `SmallIntegerField` і визначає кількість спальних місць у такому номері готелю;
- поле `bedtype`, яке має тип `CharField` і визначає тип спальних місць у такому номері готелю;
- поле `bar`, яке має тип `BooleanField` і визначає наявність бару в такому номері готелю;
- поле `tv`, яке має тип `BooleanField` і визначає наявність телебачення в такому номері готелю;
- поле `internet`, яке має тип `BooleanField` і визначає наявність інтернету в такому номері готелю;
- поле `viewtype`, яке має тип `CharField` і визначає вид, що відкривається в такому номері готелю;
- поле `balcony`, яке має тип `BooleanField` і визначає наявність балкону в такому номері готелю.

Модель фотографії готельного номеру реалізована класом `Hotelroomimage`, для реалізації якого забезпечено створення полів:

- поле `hotelroom`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає готельний номер, який показаний на фотографії;
- поле `image`, яке має тип `ImageField` і визначає фотографію, яка показує цей готельний номер.

Модель фотографії готелю реалізована класом `Hotelimage`, для реалізації якого забезпечено створення полів:

- поле `hotel`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає готель, який показаний на фотографії;
- поле `image`, яке має тип `ImageField` і визначає фотографію, яка показує цей готель.

Модель реакції клієнтів на міста, які вони відвідували, реалізована класом `Cityreaction`, для реалізації якого забезпечено створення полів:

- поле `city`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, до якого адресована реакція;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив цю реакцію;
- поле `reactiontype`, яке має тип `CharField` і визначає тип реакції;
- поле `reactionvalue`, яке має тип `SmallIntegerField` і визначає значення реакції;
- поле `reactionview`, яке має тип `CharField` і визначає текстову інформацію стосовно реакції;
- поле `reactiontime`, яке має тип `DateTimeField` і визначає час збереження реакції.

Модель реакції клієнтів на готелі, в яких вони проживали, реалізована класом `Hotelreaction`, для реалізації якого забезпечено створення полів:

- поле `hotel`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає готель, до якого адресована ця реакція;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнт, який створив цю реакцію;
- поле `reactiontype`, яке має тип `CharField` і визначає тип реакції;
- поле `reactionvalue`, яке має тип `SmallIntegerField` і визначає значення реакції;
- поле `reactionview`, яке має тип `CharField` і визначає текстову інформацію стосовно реакції;
- поле `reactiontime`, яке має тип `DateTimeField` і визначає час збереження реакції.

Модель реакції клієнтів на країни, які вони відвідували, реалізована класом `Countryreaction`, для реалізації якого забезпечено створення полів:

- поле `country`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає країну, до якої адресована ця реакція;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив цю реакцію;

- поле `reactiontype`, яке має тип `CharField` і визначає тип реакції;
- поле `reactionvalue`, яке має тип `SmallIntegerField` і визначає значення реакції;
- поле `reactionview`, яке має тип `CharField` і визначає текстову інформацію стосовно реакції;
- поле `reactiontime`, яке має тип `DateTimeField` і визначає час збереження реакції.

Модель реакції клієнтів на туристичні пакети, за якими вони подорожували, реалізована класом `Travelpackagereaction`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, до якого адресована ця реакція;
- поле `client`, яке має тип `CharField` і визначає клієнта, який створив цю реакцію;
- поле `reactiontype`, яке має тип `CharField` і визначає тип реакції;
- поле `reactionvalue`, яке має тип `SmallIntegerField` і визначає значення реакції;
- поле `reactionview`, яке має тип `CharField` і визначає текстову інформацію стосовно реакції;
- поле `reactiontime`, яке має тип `DateTimeField` і визначає час збереження реакції.

Модель закладки клієнтами міст для повернення в подальшому реалізована класом `Citybookmark`, для реалізації якого забезпечено створення полів:

- поле `city`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає місто, за сторінкою якого створено закладку;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив закладку.

Модель закладки клієнтами країн для повернення в подальшому реалізована класом `Countrybookmark`, для реалізації якого забезпечено створення полів:

- поле `country`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає країна, за сторінкою якої було створено закладку;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив закладку.

Модель закладки клієнтами готелів для повернення в подальшому реалізована класом `Hotelbookmark`, для реалізації якого забезпечено створення полів:

- поле `hotel`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає готель, за сторінкою якого було створено закладку;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив закладку.

Модель закладки клієнтами туристичних пакетів для повернення в подальшому реалізована класом `Travelpackagebookmark`, для реалізації якого забезпечено створення полів:

- поле `travelpackage`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає туристичний пакет, за сторінкою якого було створено закладку;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив закладку.

Модель закладки клієнтами туристичних рейсів для повернення в подальшому реалізована класом `Flightbookmark`, для реалізації якого забезпечено створення полів:

- поле `flight`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає рейс для переміщення з одного міста в інше, за сторінкою якого було створено закладку;
- поле `client`, яке має тип `ForeignKey` для встановлення зовнішнього ключа і визначає клієнта, який створив закладку.

4.3 Висновки за розділом 4

Визначення програмних одиниць було виконано за допомогою шаблону MVT, створюючи відповідні функції та класи, необхідні для реалізації всіх функціональних вимог до програми. Створені програмні одиниці представлені та описані.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Призначенням програми є підтримка забезпечення діяльності туристичної компанії через надання засобів для створення туристичних пакетів, визначення дат подорожей, варіантів вартості, визначення показників, отриманих від реалізації туристичного пакета, використання засобів прогнозування.

5.2 Виконання програми

Робота з програмою підтримки діяльності туристичної компанії реалізується через доступ користувачів через меню (рис. 5.1).

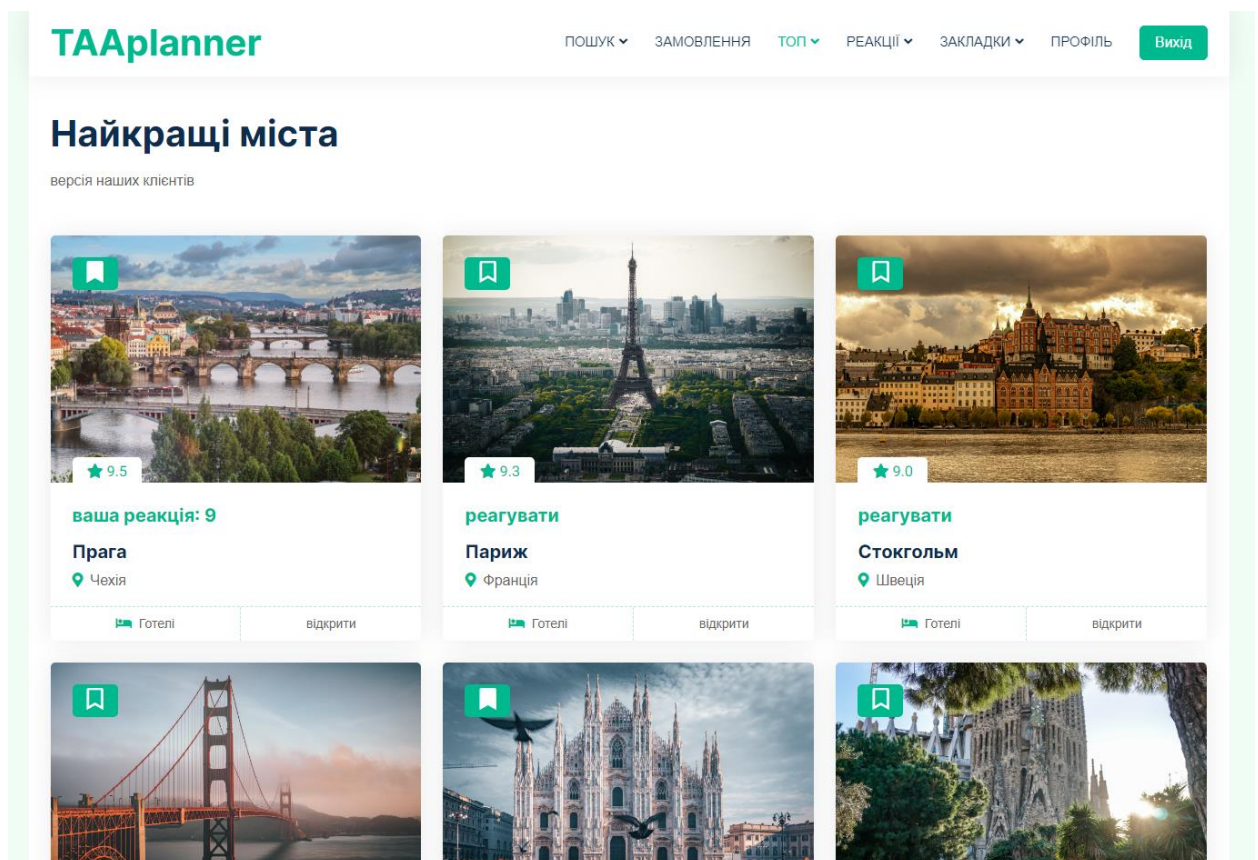


Рисунок 5.1 – Демонстрування найкращих міст за реакціями клієнтів

Розділ пошуку включає пошук міст, країн, туристичних пакетів загалом, можливих дат поїздки (туристичних пакетів на конкретну дату), готелів або кімнат в готелях, а відповідний варіант обирається з випадуючого переліку за відповідним розділом.

Розділ «Топ» містить найкращі готелі, міста, країни, туристичні пакети за реакціями, обираючи відповідний варіант з випадуючого переліку. За містами відображається посилання для переходу до готелів міста. Для повернення до країни, міста, готелю, туристичного пакету, рейсу можна створювати на них закладки. Закладки відображаються заштрихованою або незаштрихованою відповідно позначкою в лівому куті фотографії (рис. 5.1). Після того, як збереження виконано, такі закладки відображаються через розділ закладок меню з відкриттям відповідної сторінки (рис. 5.2).

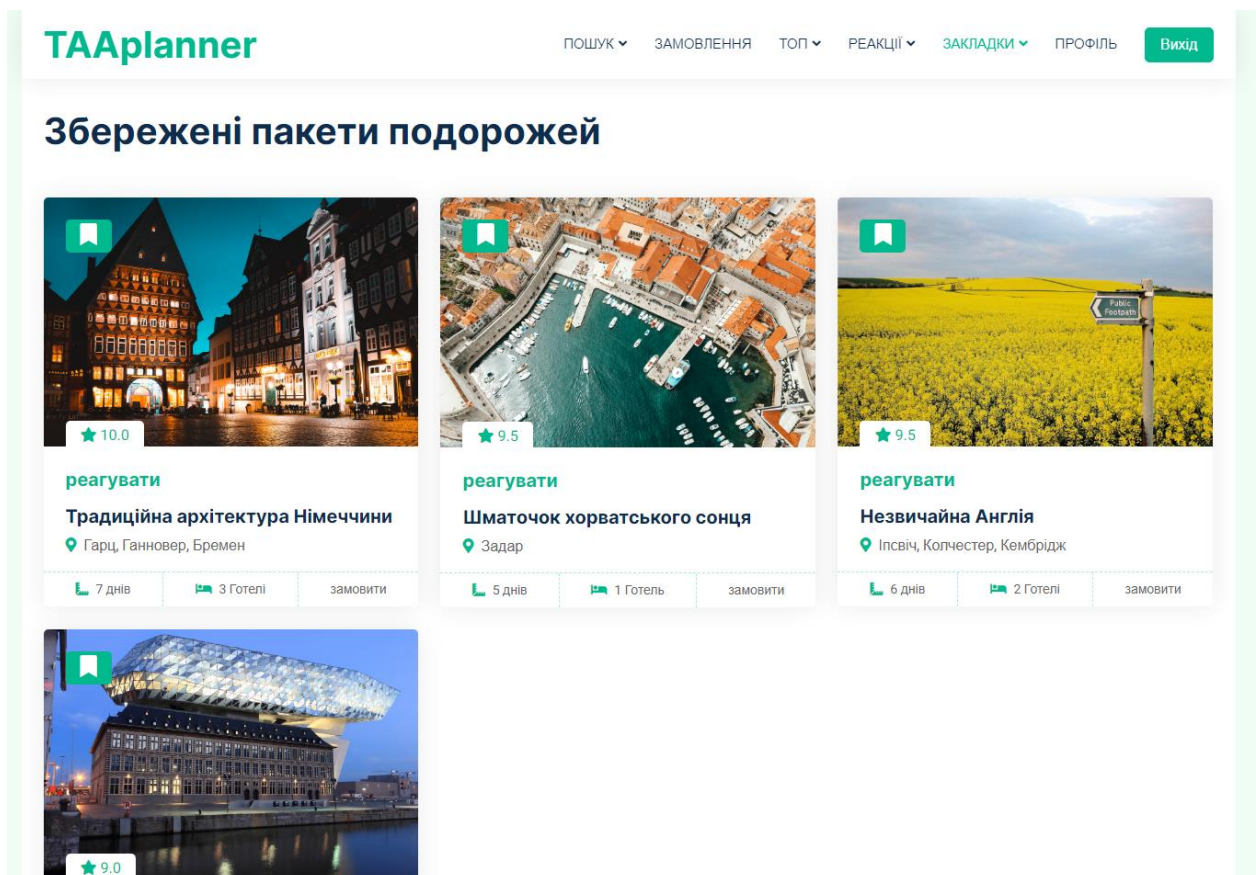


Рисунок 5.2 – Демонстрування збережених туристичних пакетів клієнтом

Також на відповідні складові (міста, готелі країни, туристичні пакети) можна реагувати. Реакція може бути виконана тільки після того, як клієнт має хоча би одне виконане замовлення. Реакція може бути або уподобанням, або оцінкою за 10-бальною шкалою. Варіант обирає сам клієнт. У випадку оцінки за 10-бальною шкалою цей варіант потім відображається при перегляді такої країни, міста, туристичного пакету клієнту. При цьому окремо демонструється середня оцінка, а окремо демонструється реакція самого цього клієнта. Якщо реакції не було, то показується посилання реагувати, але тільки за наявності виконаного замовлення.

Саме реакції клієнта можна побачити з розділу реагування в меню. Перейти ж до реакцій іншого клієнта можна, якщо дістатися його сторінки (з інформацією про нього).

Меню «Замовлення» дозволяє клієнту переглянути замовлення, які він вже створив. Для створення нового замовлення необхідно дістатися сторінки туристичного пакету з встановленою датою подорожі.

Працівник туристичної компанії не має розділів меню з реагуванням та закладок, тому що йому не доступні ці інструменти. Але він має розширений розділ з замовленнями, який дозволяє обрати потрібний йому перелік замовлень у відповідності з варіантами з функціональних вимог до програми. Звідти працівник може керувати замовленнями, підтверджуючи їх, відхиляючи або фіксуючи сплачену суму.

Також працівник туристичної компанії має розділ «Каталог», звідки можна переглядати весь список, додавати нові, редагувати існуючі міста, країни, рейси, готелі, тобто загальні дані. У свою чергу виконувати такі дії над туристичними пакетами можна з окремого меню «Пакети», звідки доступні групи туристичних пакетів, потім самі туристичні пакети, а за пакетом можна додавати дати поїздки до пакету, вартість поїздок, а також переглядати клієнтів (рис. 5.3), передивлятися дохід, який було отримано за певний період за цим пакетом. З перегляду клієнтів, що вже успішно

замовили і мають підтверджене замовлення, можна відкрити саме їх замовлення.

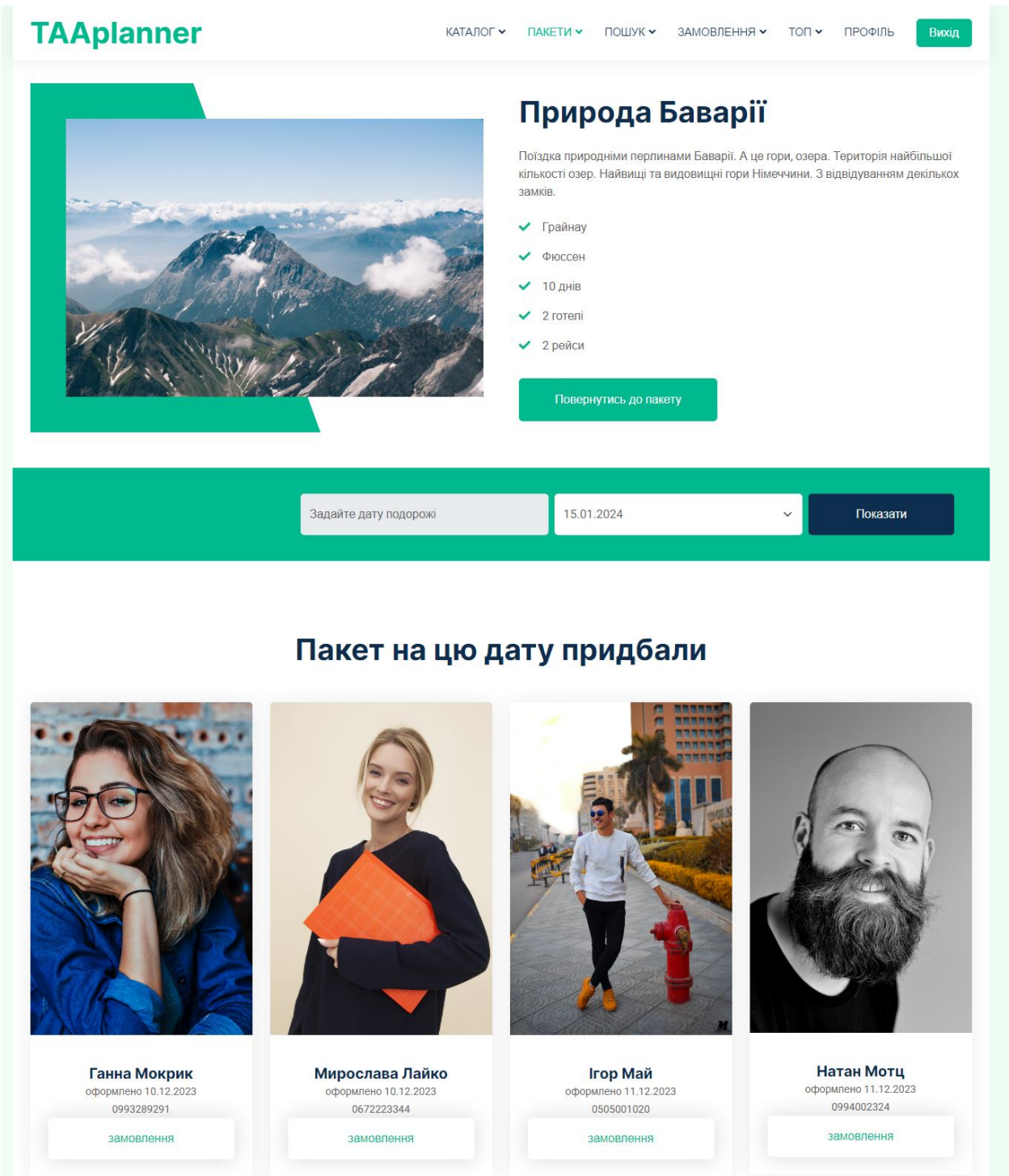
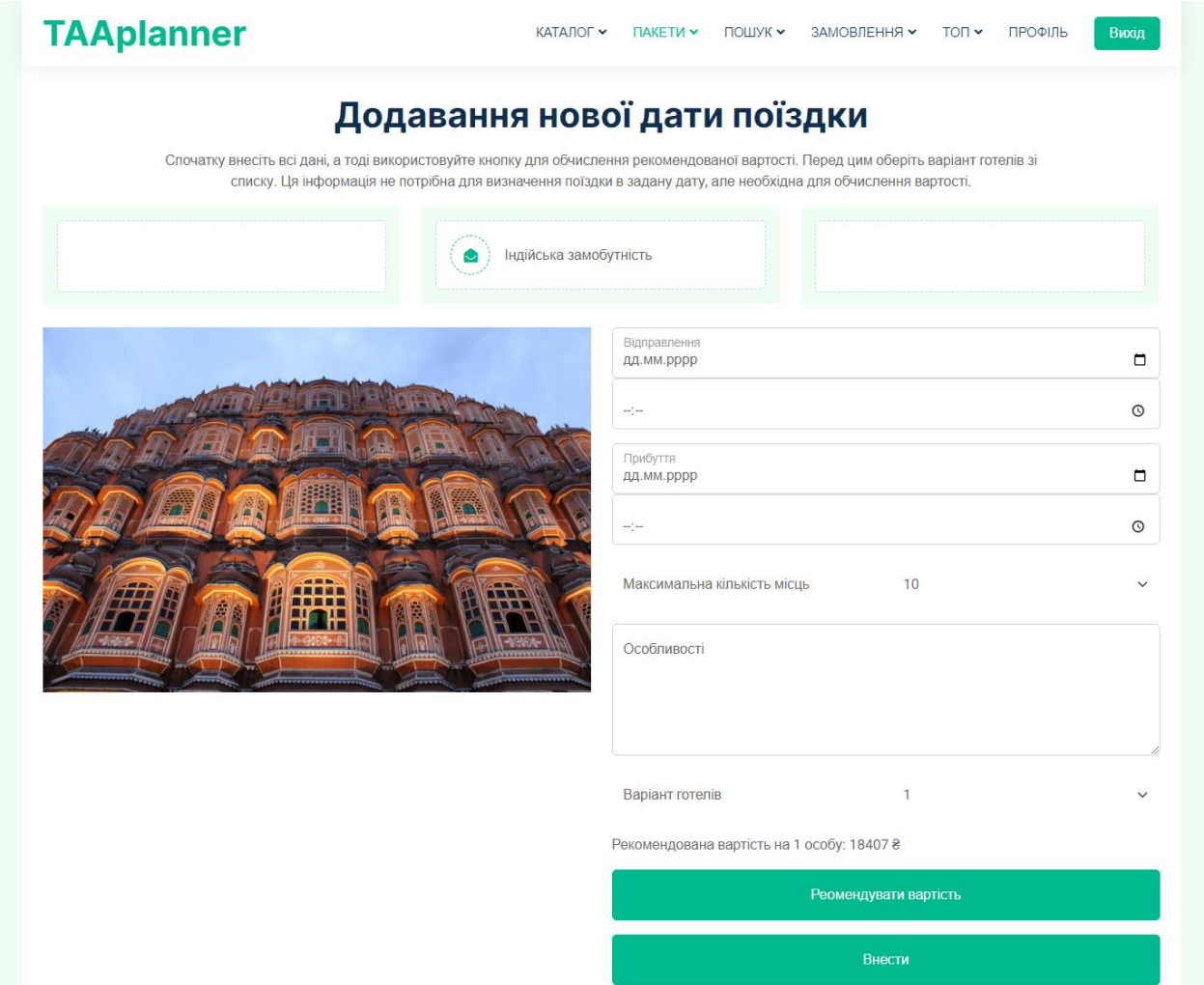


Рисунок 5.3 – Демонстрування клієнтів за датою поїздки за туристичним пакетом

Коли працівник туристичної компанії вносить дані нової поїздки за

датою, він може додатково обрати варіант готелів з доступних зі списку (якщо є альтернативні варіанти), після чого знизу відобразиться перелік цих готелів, як і рейсів, міст, де можна також деталізувати їхні дані на відповідну дату, а також з'явиться можливість натиснути на кнопку «Рекомендувати вартість», після чого буде вказано, якою є рекомендована вартість туристичного пакету на цю дату (рис. 5.4). Це значення визначається через навчену модель визначення вартості туристичного пакету.

Потім можна відповідно подивитись, якою є зараз вартість на таку дату і змінити її, якщо потрібно.



TAAplanner КАТАЛОГ ПАКЕТИ ПОШУК ЗАМОВЛЕННЯ ТОП ПРОФІЛЬ Вихід

Додавання нової дати поїздки

Спочатку внесіть всі дані, а тоді використовуйте кнопку для обчислення рекомендованої вартості. Перед цим оберіть варіант готелів зі списку. Ця інформація не потрібна для визначення поїздки в задану дату, але необхідна для обчислення вартості.

Індійська замобутність



Відправлення
дд.мм.рррр

--:--

Прибуття
дд.мм.рррр

--:--

Максимальна кількість місць 10

Особливості

Варіант готелів 1

Рекомендована вартість на 1 особу: 18407 ₪

Рекомендувати вартість

Внести

Рисунок 5.4 – Додавання нової дати поїздки за туристичним пакетом

5.3 Експериментальне дослідження

Експериментальне дослідження забезпечувалось через використання вибірки даних [12], яка знаходиться у відкритому доступі. Ця вибірка складається з 30000 записів про вартість туристичних пакетів, за кожним з яких задаючи:

- унікальний ідентифікатор;
- часову мітку збору даних;
- назву пакету;
- адресу вебсторінки;
- тип пакету;
- туристична компанія;
- місце призначення;
- план (місця призначення та кількість ночей для кожного з них);
- дата подорожі;
- перелік готелів з оцінками (не завжди представлені);
- місто початку подорожі;
- авіалінії;
- вартість за двох;
- вартість за одного;
- правила скасування.

Усі ці ознаки було додатково вивчено і досліджено. Моделі, які були створені під час дослідження, були побудовані на основі ознак, які виділені в методі визначення вартості туристичного пакета, що було запропоновано. Тому всі ці дані було оброблено описаним чином та сформовано вибірку даних, яка використовувалась вже при проведенні дослідження. Таким чином серед ознак вибірки з'явилась і тривалість подорожі за пакетом, яка достатньо сильно впливає на середню вартість туристичного пакету (рис. 5.5).

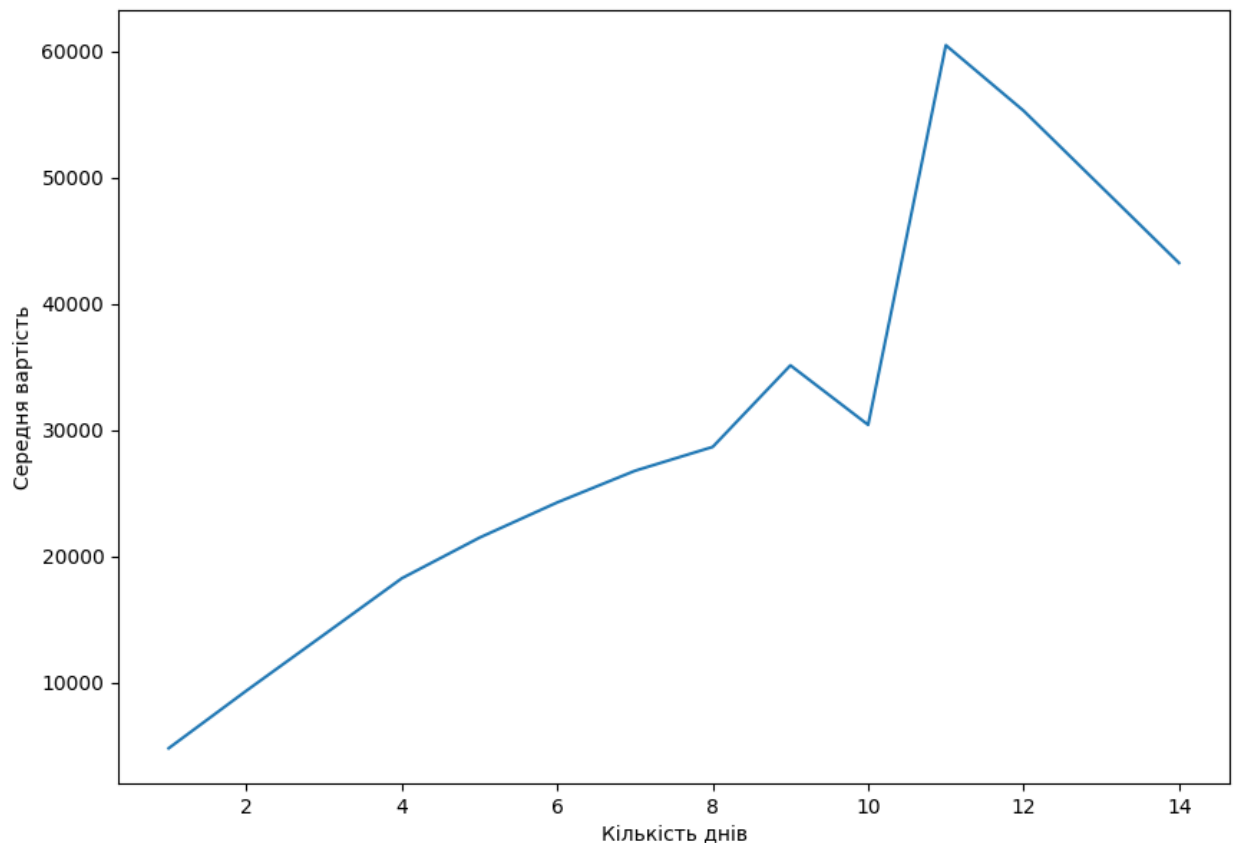


Рисунок 5.5 – Середня вартість туристичного пакету в залежності від кількості днів тривалості туристичного пакету

Було визначено кількість туристичних пакетів з кроком у 1000 індійських рупій, що відповідає такому кроку, тобто кількість пакетів вартість яких знаходиться в проміжку від заданого значення до значення на 100 більше (рис. 5.6).

Експериментальне дослідження на основі моделей прогнозування було виконано з використанням лінійної регресії, дерева рішень та екстремального градієнтного підсилювання для ансамблів дерев рішень з використанням бібліотеки XGBoost. Зі всієї вибірки 20 % даних було використано для тестування, а 80 % для навчання моделей. Результати проведеного експериментального дослідження були оцінені на основі показників коефіцієнта детермінації та Mean Absolute Error (MAE). Ці значення (табл. 5.1) були отримані тільки на основі тестової вибірки, навчальні дані при цьому не використовувались.

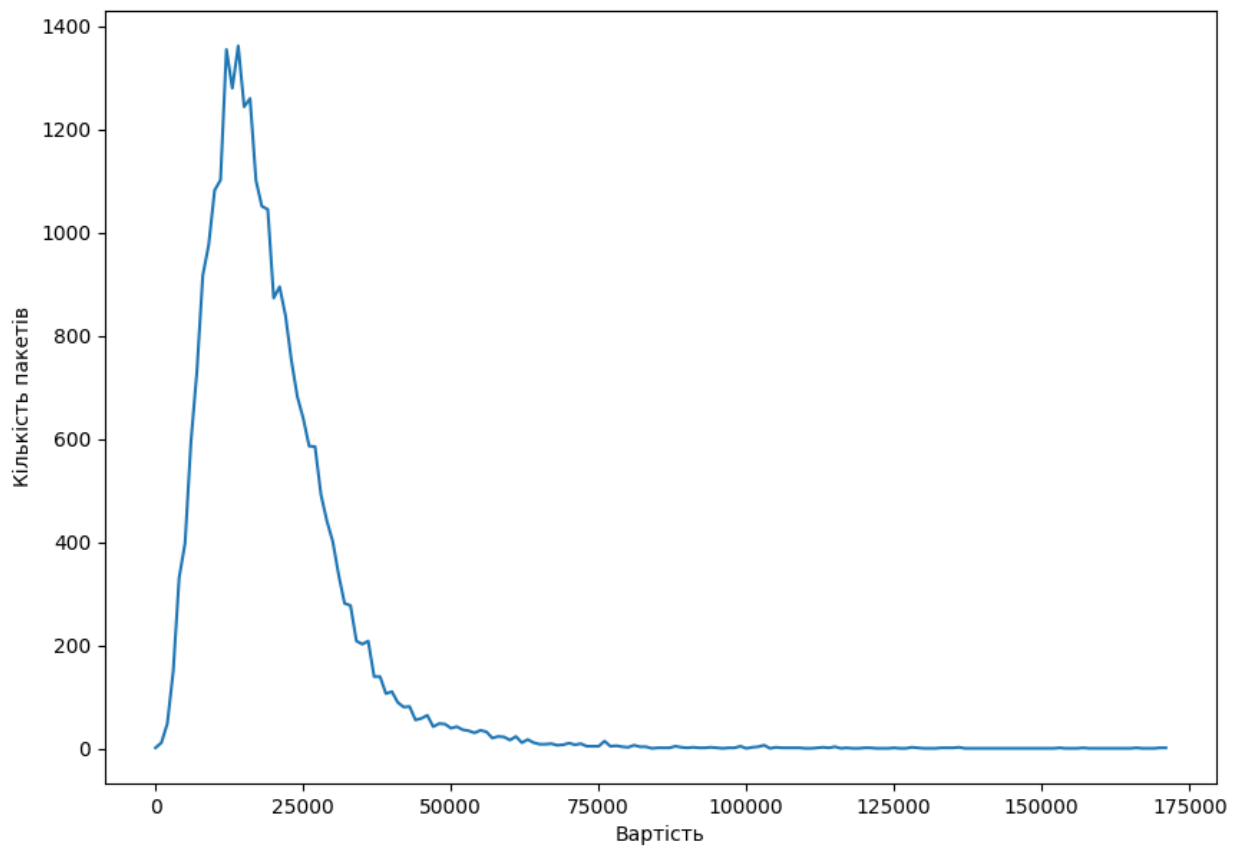


Рисунок 5.6 – Кількість туристичних пакетів з відповідною вартістю

Таблиця 5.1 – Результати проведеного експериментального дослідження

Модель	MAE	Коефіцієнт детермінації
Лінійна регресія	0,0343441	0,4422857
Дерево рішень	0,0239499	0,6733814
Екстремальне градієнтне підсилювання для ансамблів дерев рішень	0,0164359	0,806069

Найкращі результати і за MAE, і за коефіцієнтом детермінації було отримано через використання екстремального градієнтного підсилювання для ансамблів дерев рішень, модель при цьому було побудовано у

відповідності з запропонованим методом. Однак така сама логіка обробки даних була використана і в інших моделях.

Модель екстремального градієнтного підсилювання для ансамблів дерев рішень мала значення MAE на 31,4 % менше, тобто краще, за модель на основі одного окремого дерева рішень та на 52,1 % краще за лінійну регресію. При цьому коефіцієнт детермінації виявився на 82,3 % кращим за значення для моделі на основі лінійної регресії та на 19,7 % кращим за значення для дерева рішень. Тобто інформативність створеної результуючої моделі є значно вищою порівняно з іншими дослідженими моделями, що разом з покращеними значеннями MAE говорить про можливість використання на практиці створеної моделі та загалом перспективні отримані значення.

Також додатково було досліджено різні варіанти побудови такого ансамблю дерев рішень щодо впливу кількості дерев у ансамблі на отримані результати (табл. 5.2). Тож найкращі результати було отримано за наявності 100 дерев у ансамблі.

Таблиця 5.2 – Дослідження впливу кількості дерев у ансамблі на екстремальне градієнтне підсилювання для ансамблів дерев рішень

Кількість дерев у ансамблі	MAE	Коефіцієнт детермінації
10	0,0202002	0,7692323
20	0,0182589	0,7916033
40	0,0172807	0,8031989
50	0,0170289	0,8043945
100	0,0164359	0,806069
400	0,0167106	0,7899235
500	0,0168415	0,7864779

5.4 Висновки за розділом 5

Програмне забезпечення має всі реалізовані функціональні вимоги, забезпечуючи для них зокрема відповідні інтерфейси для користувачів, включає до свого складу навчену модель визначення вартості туристичних пакетів.

Результати експериментального дослідження показали, що інформативність створеної на основі методу визначення вартості туристичного пакета моделі є значно вищою порівняно з іншими дослідженими моделями, що разом з покращеними значеннями MAE говорить про можливість використання на практиці створеної моделі та загалом перспективні отримані значення. Результуюча модель мала значення MAE на 31,4 % краще за модель на основі одного окремого дерева рішень та на 52,1 % краще за лінійну регресію.

ВИСНОВКИ

Програмне забезпечення для підтримки діяльності туристичної компанії базується на використанні вбудованої моделі, яку було створено і навчено засобами програми і яка дозволяє виконувати визначення вартості туристичного пакету, що дозволяє разом з засобами моніторингу і аналізу основної діяльності компанії та її клієнтів приймати обґрунтовані управлінські рішення в підсумку.

Програма, яку було розроблено, виконує підтримку забезпечення діяльності туристичної компанії через надання засобів для створення туристичних пакетів, визначення дат подорожей, варіантів вартості, визначення показників, отриманих від реалізації туристичного пакета, використання засобів прогнозування, забезпечення зручності пошуку користувачами як через надання інструментів пошуку різних даних, так і через надання реакцій різного типу і збереження в закладках для подальшого повернення.

Наукова новизна роботи полягає в запропонованому методі визначення вартості туристичного пакета, який визначає обчислення вхідних ознак моделі на основі характеристик туристичного пакета, виконує створення моделі на основі екстремального градієнтного підсилювання для навчання ансамблів дерев рішень.

Результати експериментального дослідження показали, що інформативність створеної на основі методу визначення вартості туристичного пакета моделі є значно вищою порівняно з іншими дослідженими моделями, що разом з покращеними значеннями MAE говорить про можливість використання на практиці створеної моделі та загалом перспективні отримані значення. Результуюча модель мала значення MAE на 31,4 % краще за модель на основі одного окремого дерева рішень та на 52,1 % краще за лінійну регресію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tourism Market Industry Analysis, Size & Forecast 2032 [Electronic resource]. – Access mode : <https://www.futuremarketinsights.com/reports/tourism-market>.
2. TUI.co.uk. Holidays, Flights & Cruises 2023/2024 [Electronic resource]. – Access mode : <https://www.tui.co.uk/>.
3. Liberty Travel : Custom Vacation Packages and Travel Deals. Liberty Travel [Electronic resource]. – Access mode : <https://www.libertytravel.com/>.
4. Things to Do, Tickets, Tours & Attractions 2024. Viator [Electronic resource]. – Access mode : <https://www.viator.com/>.
5. Kanade, V. What Is Linear Regression? Types, Equation, Examples, and Best Practices for 2022 [Electronic resource] / V. Kanade. – Access mode : <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/>.
6. Hillier, W. What Is a Decision Tree and How Is It Used? [Electronic resource] / W. Hillier. – Access mode : <https://careerfoundry.com/en/blog/data-analytics/what-is-a-decision-tree/>.
7. Gradient Boosting with Intel Optimization for XGBoost [Electronic resource]. – Access mode : <https://www.intel.com/content/www/us/en/developer/articles/technical/gradient-boosting-with-xgboost.html>.
8. Nadeem, R. Flight Fare Forecasting : A Machine Learning Approach to Predict Ticket Prices [Text] / R. Nadeem, T. Sivakumar // International Conference on Data Analytics and Insights, ICDAI 2023. – Kolkata : Techno International, 2023. – P. 703-713.
9. Vageesh, D. A Machine Learning Based Approach to Predicting Flight Fares for Indian Airlines [Electronic resource] / D. Vageesh, R. Srinivasan // 2023 International Conference on Innovations in Engineering and Technology, ICIET 2023. – Muvattupuzha : St. Joseph's College of Engineering, 2023. – DOI : 10.1109/ICIET57285.2023.10220725.

10. Django [Electronic resource]. – Access mode : <https://github.com/django>.
11. Makaan – Real Estate HTML Template [Electronic resource]. – Access mode : <https://htmlcodex.com/real-estate-html-template-free/>.
12. Travel Listing From MakeMyTrip [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/promptcloud/travel-listing-from-makemytrip>.

ДОДАТОК А
Технічне завдання

Вступ

Туристичні компанії є одними з активних рушіїв економічних процесів, а тому і програмне забезпечення, яке направлене на підтримку роботи цих компаній, має бути достатньо складним, тобто містити велику кількість інструментів, для того щоби підвищити зручність роботи з даними, які генеруються під час роботи туристичної компанії, отримати з них інформацію, що може стати підґрунтям для прийняття рішень. Наявності лише форм для створення замовлень недостатньо для забезпечення ефективного набору інструментів за умов роботи в конкурентному середовищі. Тому розробка в цій роботі повинна враховувати і ґрунтуватися саме на цьому підході.

A.1 Підстави для розробки

Завдання на роботу визначено самим виданим завданням від 4 вересня 2023 року, що має входити до складу пояснювальної записки, та наказом № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка», де затверджено тему роботи: «Дослідження та програмна реалізація методів підтримки планування діяльності туристичної компанії».

A.2 Призначення розробки

Призначенням програми є підтримка забезпечення діяльності туристичної компанії через надання засобів для створення туристичних пакетів, визначення дат подорожей, варіантів вартості, визначення показників, отриманих від реалізації туристичного пакета, використання засобів прогнозування.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для того щоб досягти сформульовано призначення розробки в підсумку, потрібно виконати такі функціональні вимоги:

- відображення інформації про країну, місто, туристичний пакет, готель, кімнату в готелі;
- пошук країн, міст, туристичних пакетів, готелів, кімнат в готелі;
- пошук можливих дат поїздки;
- демонстрування всіх внесених країн, міст, груп туристичних пакетів, готелів, кімнат в готелі, рейсів;
- створення і редагування даних країн, міст, готелів, кімнат у готелі, рейсів;
- демонстрування туристичних пакетів за групою;
- замовлення туристичного пакета для подорожі;
- демонстрування власних замовлень туристичних пакетів для подорожей;
- демонстрування замовлень туристичних пакетів за працівником;
- демонстрування необроблених замовлень туристичних пакетів;
- демонстрування клієнтів за датою поїздки за туристичним пакетом, за готелем, за рейсом;
- підтвердження замовлення туристичного пакета;
- відхилення замовлення туристичного пакета;
- фіксування сплати за замовлення туристичного пакета;
- створення нового туристичного пакета, групи туристичних пакетів;
- демонстрування дат поїздки за туристичним пакетом;
- додавання, змінювання параметрів або скасування дати поїздки за туристичним пакетом;
- демонстрування варіантів вартості туристичного пакета на проміжок часу;

- додавання, змінювання варіантів вартості туристичного пакета;
- змінювання параметрів туристичного пакета, групи туристичних пакетів;
- реагування на країни, міста, готелі, туристичні пакети;
- демонстрування власних реакцій;
- демонстрування реакцій за країною, містом, готелем, туристичним пакетом;
- демонстрування реакцій клієнта за країною, містом, готелем, туристичним пакетом;
- збереження або видалення зі збережених країни, міста, готелю, туристичного пакету, рейсу до закладок;
- демонстрування країн, міст, готелів, туристичних пакетів, рейсів у власних закладках;
- демонстрування готелів за містом;
- демонстрування найкращих готелів, міст, країн, туристичних пакетів за реакціями;
- керування деталізацією туристичного пакету за конкретною датою;
- авторизація користувачів;
- демонстрування експертних пакетів туристичних подорожей;
- змінювання параметрів облікового запису;
- визначення отриманих сум за туристичним пакетом;
- визначення вартості туристичного пакета на основі прогнозування.

A.3.2 Умови експлуатації та вимоги

Умови експлуатації програми визначаються тими засобами, які мають бути обрані під час проектування програми, включаючи зокрема підтримку обраної мови програмування, організацію роботи серверу визначеним чином, забезпечення на ньому роботи системи керування базами даних, яку має бути обрано, а також будь-яких додаткових засобів, які обрані та використані під

час розробки програми та потребують інсталяції.

A.4 Порядок контролю та приймання

Контроль виконання заявлених складових роботи має відбуватися згідно з завданням, виданим на роботу, що є однією з підстав для розробки. Виконання роботи має бути завершено протягом 12 тижнів з проведенням захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from django.shortcuts import render
from .models import Country
from .models import City
from .models import Travelpackagecity
from .models import Cityimage
from .models import Countryimage
from .models import Hotel
from .models import Hotelroom
from .models import Hotelimage
from .models import Travelpackagehotelroom
from .models import Client
from .models import Countryreaction
from .models import Countrybookmark
from .models import Cityreaction
from .models import Citybookmark
from .models import Travelpackagebookmark
from .models import Travelpackagereaction
from .models import Expert
from .models import Experttravelpackage
from .models import Flightbookmark
from .models import Hotelbookmark
from .models import Hotelreaction

def countrypageview(request, countrytitle):
    try:
        country_data = Country.objects.get(title = countrytitle)
    except Country.DoesNotExist:
        return render(request, "taaplanner/country.html", {})
    countryimages = Countryimage.objects.filter(country =
country_data)
    countrycities = City.objects.filter(country = country_data)
    countrycities = countrycities.order_by("-
reaction").values()[:10]
    travelpackages =
Travelpackagecity.objects.filter(city__country = country_data)
    travelpackages =
travelpackages.select_related("travelpackage")
    travelpackages = travelpackages.order_by("-
travelpackage__reaction")
    travelpackages = travelpackages.values()[:10]
    try:
        client_data = Client.objects.get(user = request.user)
        reactions = Countryreaction.objects.filter(country =
country_data, client = client_data)
        if reactions and reactions.count() > 0:
            reaction = True
        else:
            reaction = False

```

```

        bookmarks = Countrybookmark.objects.filter(country =
country_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            bookmark = True
        else:
            bookmark = False
    except Client.DoesNotExist:
        bookmark = False
        reaction = False
        client_data = None
    if client_data:
        for travelpack in travelpackages:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = travelpack,
client = client_data)
            if reactions and reactions.count() > 0:
                travelpack.reaction = True
            else:
                travelpack.reaction = False
            bookmarks =
Travelpackagebookmark.objects.filter(travelpackage = travelpack,
client = client_data)
            if bookmarks and bookmarks.count() > 0:
                travelpack.bookmark = True
            else:
                travelpack.bookmark = False
        for countrycity in countrycities:
            reactions = Cityreaction.objects.filter(city =
countrycity, client = client_data)
            if reactions and reactions.count() > 0:
                countrycity.reaction = True
            else:
                countrycity.reaction = False
            bookmarks = Citybookmark.objects.filter(city =
countrycity, client = client_data)
            if bookmarks and bookmarks.count() > 0:
                countrycity.bookmark = True
            else:
                countrycity.bookmark = False
    return render(request, "taaplanner/country.html",
{"country": country_data,
"images": countryimages,
"cities": countrycities,
"packages": travelpackages,
"reaction": reaction,
"bookmark": bookmark })

def citypageview(request, citytitle):
    try:

```

```

        city_data = City.objects.select_related("country")
        city_data = city_data.get(title = citytitle)
    except City.DoesNotExist:
        return render(request, "taaplanner/city.html", {})
    country_data = city_data.country
    travelpackages = Travelpackagecity.objects.filter(city =
city_data)
    travelpackages =
travelpackages.select_related("travelpackage")
    travelpackages = travelpackages.order_by("-
travelpackage__reaction")
    travelpackages = travelpackages.values()[:10]
    cityimages = Cityimage.objects.filter(city = city_data)
    try:
        client_data = Client.objects.get(user = request.user)
        reactions = Cityreaction.objects.filter(city= city_data,
client = client_data)
        if reactions and reactions.count() > 0:
            reaction = True
        else:
            reaction = False
        bookmarks = Citybookmark.objects.filter(city =
city_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            bookmark = True
        else:
            bookmark = False
    except Client.DoesNotExist:
        bookmark = False
        reaction = False
        client_data = None
    if client_data:
        for travelpack in travelpackages:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = travelpack,
client = client_data)
            if reactions and reactions.count() > 0:
                travelpack.reaction = True
            else:
                travelpack.reaction = False
            bookmarks =
Travelpackagebookmark.objects.filter(travelpackage = travelpack,
client = client_data)
            if bookmarks and bookmarks.count() > 0:
                travelpack.bookmark = True
            else:
                travelpack.bookmark = False
            reactions = Countryreaction.objects.filter(country =
country_data, client = client_data)

```

```

        if reactions and reactions.count() > 0:
            country_reaction = True
        else:
            country_reaction = False
        bookmarks = Countrybookmark.objects.filter(country =
country_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            country_bookmark = True
        else:
            country_bookmark = False
    return render(request, "taaplanner/city.html",
        {"city": city_data,
         "images": cityimages,
         "packages": travelpackages,
         "reaction": reaction,
         "bookmark": bookmark,
         "country_reaction": country_reaction,
         "country_bookmark": country_bookmark })

def hotelpageview(request, hotelslug):
    try:
        hotel_data = Hotel.objects.select_related("city")
        hotel_data = hotel_data.select_related("city__country")
        hotel_data = hotel_data.get(title = hotelslug)
    except Hotel.DoesNotExist:
        return render(request, "taaplanner/hotel.html", {})
    city_data = hotel_data.city
    country_data = hotel_data.city.country
    hotelrooms = Hotelroom.objects.filter(hotel = hotel_data)
    hotelrooms = hotelrooms.select_related("hotel")
    hotelimages = Hotelimage.objects.filter(hotel = hotel_data)
    travelpackages =
Travelpackagehotelroom.objects.filter(hotelroom__hotel = hotel_data)
    travelpackages =
travelpackages.select_related("travelpackage")
    travelpackages = travelpackages.order_by("-
travelpackage__reaction").values()[:10]
    try:
        client_data = Client.objects.get(user = request.user)
        reactions = Hotelreaction.objects.filter(hotel=
hotel_data, client = client_data)
        if reactions and reactions.count() > 0:
            reaction = True
        else:
            reaction = False
        bookmarks = Hotelbookmark.objects.filter(hotel=
hotel_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            bookmark = True

```

```

        else:
            bookmark = False
    except Client.DoesNotExist:
        bookmark = False
        reaction = False
        client_data = None
    if client_data:
        for travelpack in travelpackages:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = travelpack,
client = client_data)
            if reactions and reactions.count() > 0:
                travelpack.reaction = True
            else:
                travelpack.reaction = False
            bookmarks =
Travelpackagebookmark.objects.filter(travelpackage = travelpack,
client = client_data)
            if bookmarks and bookmarks.count() > 0:
                travelpack.bookmark = True
            else:
                travelpack.bookmark = False
        reactions = Countryreaction.objects.filter(country =
country_data, client = client_data)
        if reactions and reactions.count() > 0:
            country_reaction = True
        else:
            country_reaction = False
        bookmarks = Countrybookmark.objects.filter(country =
country_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            country_bookmark = True
        else:
            country_bookmark = False
        reactions = Cityreaction.objects.filter(city =
city_data, client = client_data)
        if reactions and reactions.count() > 0:
            city_reaction = True
        else:
            city_reaction = False
        bookmarks = Citybookmark.objects.filter(city =
city_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            city_bookmark = True
        else:
            city_bookmark = False
    return render(request, "taaplanner/hotel.html",
        {"hotel": hotel_data,
         "images": hotelimages,

```

```

        "rooms": hotelrooms,
        "packages": travelpackages,
        "country_reaction": country_reaction,
        "country_bookmark": country_bookmark,
        "city_reaction": city_reaction,
        "city_bookmark": city_bookmark })

def hotelroompageview(request, hotelroom):
    try:
        hotelroom_data =
Hotelroom.objects.select_related("hotel")
        hotelroom_data =
hotelroom_data.select_related("hotel__city")
        hotelroom_data =
hotelroom_data.select_related("hotel__city__country")
        hotelroom_data = hotelroom_data.get(id = hotelroom)
    except Hotelroom.DoesNotExist:
        return render(request, "taaplanner/hotelroom.html", {})
    city_data = hotelroom_data.hotel.city
    country_data = hotelroom_data.hotel.city.country
    hotelroomimages = Hotelroomimage.objects.filter(hotelroom =
hotelroom_data)
    travelpackages =
Travelpackagehotelroom.objects.filter(hotelroom = hotelroom)
    travelpackages =
travelpackages.select_related("travelpackage")
    travelpackages = travelpackages.order_by("-
travelpackage__reaction")
    travelpackages = travelpackages.distinct()
    travelpackages = travelpackages.values()[:10]
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        reactions = Countryreaction.objects.filter(country =
country_data, client = client_data)
        if reactions and reactions.count() > 0:
            country_reaction = True
        else:
            country_reaction = False
        bookmarks = Countrybookmark.objects.filter(country =
country_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            country_bookmark = True
        else:
            country_bookmark = False
        reactions = Cityreaction.objects.filter(city =
city_data, client = client_data)

```

```

        if reactions and reactions.count() > 0:
            city_reaction = True
        else:
            city_reaction = False
        bookmarks = Citybookmark.objects.filter(city =
city_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            city_bookmark = True
        else:
            city_bookmark = False
    return render(request, "taaplanner/hotelroom.html",
        {"hotelroom": hotelroom_data,
        "images": hotelroomimages,
        "packages": travelpackages,
        "country_reaction": country_reaction,
        "country_bookmark": country_bookmark,
        "city_reaction": city_reaction,
        "city_bookmark": city_bookmark })

def travelpackagepageview(request, travelpackageslug):
    try:
        travelpackage_data =
Travelpackage.objects.select_related("travelpackagegroup")
        travelpackage_data = travelpackage_data.get(slug =
hotelslug)
    except Travelpackage.DoesNotExist:
        return render(request, "taaplanner/package.html", {})
    packagecities =
Travelpackagecity.objects.filter(travelpackage = travelpackage_data)
    packagecities = packagecities.select_related("city")
    packagecities =
packagecities.select_related("city__country")
    packageflights =
Travelpackageflight.objects.filter(travelpackage = travelpackage_data)
    packageflights = packageflights.select_related("sourcecity")
    packageflights =
packageflights.select_related("sourcecity__country")
    packageflights =
packageflights.select_related("destinationcity")
    packageflights =
packageflights.select_related("destinationcity__country")
    packageflights = packageflights.select_related("flight")
    packageflights =
packageflights.select_related("flight__company")
    packageflights =
packageflights.select_related("flight__sourcecity")
    packageflights =
packageflights.select_related("flight__sourcecity__country")

```

```

        packageflights =
packageflights.select_related("flight__destinationcity")
        packageflights =
packageflights.select_related("flight__destinationcity__country")
        packagehotels =
Travelpackagehotelroom.objects.filter(travelpackage =
travelpackage_data)
        packagehotels = packagehotels.select_related("hotelroom")
        packagehotels =
packagehotels.select_related("hotelroom__hotel")
        packagehotels =
packagehotels.select_related("hotelroom__hotel__city")
        packagehotels =
packagehotels.select_related("hotelroom__hotel__city__country")
    try:
        client_data = Client.objects.get(user = request.user)
        reactions =
Travelpackagereaction.objects.filter(travelpackage=travelpackage_data,
client=client_data)
        if reactions and reactions.count() > 0:
            reaction = True
        else:
            reaction = False
        bookmarks =
Travelpackagebookmark.objects.filter(travelpackage=
travelpackage_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            bookmark = True
        else:
            bookmark = False
    except Client.DoesNotExist:
        bookmark = False
        reaction = False
        client_data = None
    if client_data:
        for packcity in packagecities:
            reactions = Cityreaction.objects.filter(city =
packcity, client = client_data)
            if reactions and reactions.count() > 0:
                packcity.reaction = True
            else:
                packcity.reaction = False
            bookmarks = Citybookmark.objects.filter(city =
packcity, client = client_data)
            if bookmarks and bookmarks.count() > 0:
                packcity.bookmark = True
            else:
                packcity.bookmark = False
        for packflight in packageflights:

```

```

        bookmarks = Flightbookmark.objects.filter(flight =
packflight, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            packflight.bookmark = True
        else:
            packflight.bookmark = False
    for packhotel in packagehotels:
        reactions = Hotelreaction.objects.filter(hotel =
packhotel, client = client_data)
        if reactions and reactions.count() > 0:
            packhotel.reaction = True
        else:
            packhotel.reaction = False
        bookmarks = Hotelbookmark.objects.filter(hotel =
packhotel, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            packhotel.bookmark = True
        else:
            packhotel.bookmark = False
    return render(request, "taaplanner/package.html",
{"package": travelpackage_data,
"reaction": reaction,
"bookmark": bookmark,
"cities": packagecities,
"flights": packageflights,
"hotels": packagehotels })

def employeecountriespageview(request):
    employeecountries = Country.objects.all()
    employeecountries = employeecountries.order_by("title")
    return render(request, "taaplanner/countries.html",
{"countries": employeecountries })

def employeecitiespageview(request):
    employeecities = City.objects.all()
    employeecities = employeecities.select_related("country")
    employeecities = employeecities.order_by("title")
    return render(request, "taaplanner/cities.html",
{"cities": employeecities })

def employeetravelpackagegroupspageview(request):
    employeetravelpackagegroups =
Travelpackagegroup.objects.all()
    employeetravelpackagegroups =
employeetravelpackagegroups.order_by("title")
    return render(request, "taaplanner/travelgroups.html",
{"packages": employeetravelpackagegroups })

def employeetravelpackagespageview(request):

```

```

        employeetravelpackages = Travelpackage.objects.all()
        employeetravelpackages =
employeetravelpackages.order_by("title")
        return render(request, "taaplanner/travelpackages.html",
            {"packages": employeetravelpackages })

    def employeehotelspageview(request):
        employeehotels = Hotel.objects.all()
        employeehotels = employeehotels.order_by("title")
        return render(request, "taaplanner/hotels.html", {"hotels":
employeehotels })

    def employeehotelroomspageview(request):
        employeehotelrooms = Hotelroom.objects.all()
        employeehotelrooms = employeehotelrooms.order_by("title")
        return render(request, "taaplanner/rooms.html", {"rooms":
employeehotelrooms })

    def employeeflightspageview(request):
        employeeflights = Flight.objects.all()
        employeeflights = employeeflights.order_by("id")
        return render(request, "taaplanner/flights.html",
{"flights": employeeflights })

    def grouptravelpackagesview(request, travelpackagegroupslug):
        try:
            grouptravelpackage_data =
Travelpackagegroup.objects.get(slug = travelpackagegroupslug)
        except Travelpackagegroup.DoesNotExist:
            return render(request, "taaplanner/packagehroup.html",
{})

        grouppackages =
Travelpackage.objects.filter(travelpackagegroup =
grouptravelpackage_data)
        try:
            client_data = Client.objects.get(user = request.user)
        except Client.DoesNotExist:
            client_data = None
        if client_data:
            for travelpack in grouppackages:
                reactions =
Travelpackagereaction.objects.filter(travelpackage = travelpack,
client = client_data)
                if reactions and reactions.count() > 0:
                    travelpack.reaction = True
                else:
                    travelpack.reaction = False

```

```

        bookmarks =
    Travelpackagebookmark.objects.filter(travelpackage = travelpack,
    client = client_data)
        if bookmarks and bookmarks.count() > 0:
            travelpack.bookmark = True
        else:
            travelpack.bookmark = False
    return render(request,
    "taaplanner/grouptravelpackages.html",
        {"group": grouptravelpackage_data,
        "packages": grouppackages })

def cityhotelspageview(request, citytitle):
    try:
        city_data = City.objects.select_related("country")
        city_data = city_data.get(title = citytitle)
    except City.DoesNotExist:
        return render(request, "taaplanner/cityhotels.html", {})
    country_data = city_data.country
    cityhotels = Hotel.objects.filter(city = city_data)
    cityhotels = cityhotels.order_by("-reaction")
    try:
        client_data = Client.objects.get(user = request.user)
        reactions = Cityreaction.objects.filter(city =
city_data, client = client_data)
        if reactions and reactions.count() > 0:
            reaction = True
        else:
            reaction = False
        bookmarks = Citybookmark.objects.filter(city =
city_data, client = client_data)
        if bookmarks and bookmarks.count() > 0:
            bookmark = True
        else:
            bookmark = False
    except Client.DoesNotExist:
        bookmark = False
        reaction = False
        client_data = None
    if client_data:
        for cityhotel in cityhotels:
            reactions = Hotelreaction.objects.filter(hotel =
cityhotel, client = client_data)
            if reactions and reactions.count() > 0:
                cityhotel.reaction = True
            else:
                cityhotel.reaction = False
            bookmarks = Hotelbookmark.objects.filter(hotel =
cityhotel, client = client_data)

```

```

        if bookmarks and bookmarks.count() > 0:
            cityhotel.bookmark = True
        else:
            cityhotel.bookmark = False
    return render(request, "taaplanner/cityhotels.html",
        {"city": city_data,
         "hotels": cityhotels,
         "reaction": reaction,
         "bookmark": bookmark })

def besthotelspageview(request):
    cityhotels = Hotel.objects.all()
    cityhotels = cityhotels.select_related("city")
    cityhotels = cityhotels.select_related("city__country")
    cityhotels = cityhotels.order_by("-reaction")
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        for cityhotel in cityhotels:
            reactions = Hotelreaction.objects.filter(hotel =
cityhotel, client = client_data)
            if reactions and reactions.count() > 0:
                cityhotel.reaction = True
            else:
                cityhotel.reaction = False
            bookmarks = Hotelbookmark.objects.filter(hotel =
cityhotel, client = client_data)
            if bookmarks and bookmarks.count() > 0:
                cityhotel.bookmark = True
            else:
                cityhotel.bookmark = False
    return render(request, "taaplanner/cityhotels.html",
        {"hotels": cityhotels })

def bestcountriespageview(request):
    bestcountries = Country.objects.all()
    bestcountries = bestcountries.order_by("-reaction")
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        for becountr in bestcountries:
            reactions = Countryreaction.objects.filter(country =
becountr, client = client_data)
            if reactions and reactions.count() > 0:
                becountr.reaction = True

```

```

        else:
            becountry.reaction = False
            bookmarks = Countrybookmark.objects.filter(country =
becountry, client = client_data)
            if bookmarks and bookmarks.count() > 0:
                becountry.bookmark = True
            else:
                becountry.bookmark = False
    return render(request, "taaplanner/bestcountries.html",
        {"countries": bestcountries })

def bestcitiespageview(request):
    bestcities = City.objects.all()
    bestcities = bestcities.select_related("country")
    bestcities = bestcities.order_by("-reaction")
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        for becity in bestcities:
            reactions = Cityreaction.objects.filter(city =
becity, client = client_data)
            if reactions and reactions.count() > 0:
                becity.reaction = True
            else:
                becity.reaction = False
            bookmarks = Citybookmark.objects.filter(city =
becity, client = client_data)
            if bookmarks and bookmarks.count() > 0:
                becity.bookmark = True
            else:
                becity.bookmark = False
    return render(request, "taaplanner/bestcities.html",
        {"cities": bestcities })

def besttravelpackagespageview(request):
    besttravpackages = Travelpackage.objects.all()
    besttravpackages =
besttravpackages.select_related("travelpackagegroup")
    besttravpackages = besttravpackages.order_by("-reaction")
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        for travpackage in besttravpackages:

```

```

        reactions =
Travelpackagereaction.objects.filter(travelpackage = travpackage,
client = client_data)
        if reactions and reactions.count() > 0:
            travpackage.reaction = True
        else:
            travpackage.reaction = False
        bookmarks =
Travelpackagebookmark.objects.filter(travelpackage = travpackage,
client = client_data)
        if bookmarks and bookmarks.count() > 0:
            travpackage.bookmark = True
        else:
            travpackage.bookmark = False
        travpackage.cities =
Travelpackagecity.objects.filter(travelpackage =
travpackage).select_related("city").order_by("startday")
        return render(request, "taaplanner/bestpackages.html",
            {"packages": besttravpackages })

def expertpackagespageview(request, expertname):
    try:
        expert_data = Expert.objects.select_related("country")
        expert_data = expert_data.get(user__username =
expertname)
    except Expert.DoesNotExist:
        return render(request, "taaplanner/expertpackages.html",
        {})

    experttravpackages =
Experttravelpackage.objects.filter(expert = expert_data)
    experttravpackages =
experttravpackages.select_related("travelpackage")
    experttravpackages = experttravpackages.order_by("-
travelpackage__reaction")
    experttravpackages =
experttravpackages.values_list("travelpackage")
    experttravpackages = experttravpackages.distinct()
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        for travpackage in experttravpackages:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = travpackage,
client = client_data)
            if reactions and reactions.count() > 0:
                travpackage.reaction = True
            else:

```

```

        travpackage.reaction = False
        bookmarks =
    Travelpackagebookmark.objects.filter(travpackage = travpackage,
    client = client_data)
        if bookmarks and bookmarks.count() > 0:
            travpackage.bookmark = True
        else:
            travpackage.bookmark = False
    return render(request, "taaplanner/expertpackages.html",
        {"packages": experttravpackages })

def bookmarkedcountriespageview(request):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        bookmarkedcountries =
Countrybookmark.objects.filter(client = client_data)
        bookmarkedcountries =
bookmarkedcountries.select_related("country")
        bookmarkedcountries =
bookmarkedcountries.values_list("country")
        for bkcountry in bookmarkedcountries:
            reactions = Countryreaction.objects.filter(country =
bkcountry, client = client_data)
            if reactions and reactions.count() > 0:
                bkcountry.reaction = True
            else:
                bkcountry.reaction = False
    return render(request,
"taaplanner/bookmarkedcountries.html",
        {"countries": bookmarkedcountries })

def bookmarkedcitiespageview(request):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        bookmarkedcities = Citybookmark.objects.filter(client =
client_data)
        bookmarkedcities =
bookmarkedcities.select_related("city")
        bookmarkedcities =
bookmarkedcities.select_related("city__country")
        bookmarkedcities = bookmarkedcities.values_list("city")
        for bkcity in bookmarkedcities:

```

```

        reactions = Cityreaction.objects.filter(city =
bkcity, client = client_data)
        if reactions and reactions.count() > 0:
            bkcity.reaction = True
        else:
            bkcity.reaction = False
    return render(request, "taaplanner/bookmarkedcities.html",
        {"countries": bookmarkedcities })

def bookmarkedhotelspageview(request):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        bookmarkedhotels = Hotelbookmark.objects.filter(client =
client_data)
        bookmarkedhotels =
bookmarkedhotels.select_related("hotel")
        bookmarkedhotels =
bookmarkedhotels.select_related("hotel__city")
        bookmarkedhotels =
bookmarkedhotels.select_related("hotel__city__country")
        bookmarkedhotels = bookmarkedhotels.values_list("hotel")
        for bkhotel in bookmarkedhotels:
            reactions = Hotelreaction.objects.filter(hotel =
bkhotel, client = client_data)
            if reactions and reactions.count() > 0:
                bkhotel.reaction = True
            else:
                bkhotel.reaction = False
    return render(request, "taaplanner/bookmarkedhotels.html",
        {"hotels": bookmarkedhotels})

def bookmarkedflightspageview(request):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        bookmarkedflights = Flightbookmark.objects.filter(client
= client_data)
        bookmarkedflights =
bookmarkedflights.select_related("flight")
        bookmarkedflights =
bookmarkedflights.select_related("company")
        bookmarkedflights =
bookmarkedflights.select_related("sourcecity")

```

```

        bookmarkedflights =
bookmarkedflights.select_related("destinationcity")
        bookmarkedflights =
bookmarkedflights.select_related("sourcecity__country")
        bookmarkedflights =
bookmarkedflights.select_related("destinationcity__country")
        bookmarkedflights =
bookmarkedflights.values_list("flight")
        return render(request, "taaplanner/bookmarkedflights.html",
                        {"flights": bookmarkedflights })

def bookmarkedpackagespageview(request):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    if client_data:
        bookmarkedpackages =
Travelpackagebookmark.objects.filter(client = client_data)
        bookmarkedpackages =
bookmarkedpackages.select_related("travelpackage")
        bookmarkedpackages =
bookmarkedpackages.values_list("travelpackage")
        for bkpackage in bookmarkedpackages:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = bkpackage, client
= client_data)
            if reactions and reactions.count() > 0:
                bkpackage.reaction = True
            else:
                bkpackage.reaction = False
        return render(request, "taaplanner/bookmarkedpackages.html",
                        {"packages": bookmarkedpackages })

def clientreactionspageview(request, clientname = None):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    try:
        if clientname:
            reaction_client_data =
Client.objects.get(user__username = clientname)
        else:
            reaction_client_data = Client.objects.get(user =
request.user)
    except Client.DoesNotExist:
        client_data = None

```

```

        package_reactions =
    Travelpackagereaction.objects.filter(client = reaction_client_data)
        package_reactions =
    package_reactions.select_related("travelpackage")
        country_reactions = Countryreaction.objects.filter(client =
    reaction_client_data)
        country_reactions =
    country_reactions.select_related("country")
        hotel_reactions = Hotelreaction.objects.filter(client =
    reaction_client_data)
        hotel_reactions = hotel_reactions.select_related("hotel")
        city_reactions = Cityreaction.objects.filter(client =
    reaction_client_data)
        city_reactions = city_reactions.select_related("city")
        city_reactions = city_reactions.select_related("country")
        if client_data:
            for bkpackage in package_reactions:
                reactions =
    Travelpackagereaction.objects.filter(travelpackage = bkpackage, client
    = client_data)
                if reactions and reactions.count() > 0:
                    bkpackage.reaction = True
                else:
                    bkpackage.reaction = False
                bookmarks =
    Travelpackagebookmark.objects.filter(travelpackage = bkpackage, client
    = client_data)
                if reactions and reactions.count() > 0:
                    bkpackage.bookmark = True
                else:
                    bkpackage.bookmark = False
            for bkhotel in hotel_reactions:
                reactions = Hotelreaction.objects.filter(hotel =
    bkhotel, client = client_data)
                if reactions and reactions.count() > 0:
                    bkhotel.reaction = True
                else:
                    bkhotel.reaction = False
                bookmarks = Hotelbookmark.objects.filter(hotel =
    bkhotel, client = client_data)
                if reactions and bkhotel.count() > 0:
                    bkhotel.bookmark = True
                else:
                    bkhotel.bookmark = False
            for bkcity in city_reactions:
                reactions = Cityreaction.objects.filter(city =
    bkcity, client = client_data)
                if reactions and reactions.count() > 0:
                    bkcity.reaction = True

```

```

        else:
            bkcity.reaction = False
            bookmarks = Citybookmark.objects.filter(city =
bkcity, client = client_data)
            if reactions and bkhotel.count() > 0:
                bkcity.bookmark = True
            else:
                bkcity.bookmark = False
        for bkcountry in country_reactions:
            reactions = Countryreaction.objects.filter(country =
bkcountry, client = client_data)
            if reactions and reactions.count() > 0:
                bkcountry.reaction = True
            else:
                bkcountry.reaction = False
            bookmarks = Countrybookmark.objects.filter(country =
bkcountry, client = client_data)
            if reactions and bkhotel.count() > 0:
                bkcountry.bookmark = True
            else:
                bkcountry.bookmark = False
    return render(request, "taaplanner/clientreactions.html",
{"packages": package_reactions,
"hotels": hotel_reactions,
"countries": country_reactions,
"cities": city_reactions })

def countryreactionspageview(request, countrytitle):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    try:
        country_data = Country.objects.get(title = countrytitle)
    except Country.DoesNotExist:
        return render(request,
"taaplanner/countryreactions.html", {})
    country_reactions = Countryreaction.objects.filter(country =
country_data)
    if client_data:
        for bkcountry in country_reactions:
            reactions = Countryreaction.objects.filter(country =
bkcountry, client = client_data)
            if reactions and reactions.count() > 0:
                bkcountry.reaction = True
            else:
                bkcountry.reaction = False
            bookmarks = Countrybookmark.objects.filter(country =
bkcountry, client = client_data)

```

```

        if reactions and bkhotel.count() > 0:
            bkcountry.bookmark = True
        else:
            bkcountry.bookmark = False
    return render(request, "taaplanner/countryreactions.html",
                  {"countries": country_reactions })

def cityreactionspageview(request, citytitle):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    try:
        city_data = City.objects.get(title = citytitle)
    except City.DoesNotExist:
        return render(request, "taaplanner/cityreactions.html",
                      {"cities": [] })
    city_reactions = Cityreaction.objects.filter(city =
city_data)
    city_reactions = city_reactions.select_related("city")
    city_reactions = city_reactions.select_related("country")
    if client_data:
        for bkcity in city_reactions:
            reactions = Cityreaction.objects.filter(city =
bkcity, client = client_data)
            if reactions and reactions.count() > 0:
                bkcity.reaction = True
            else:
                bkcity.reaction = False
            bookmarks = Citybookmark.objects.filter(city =
bkcity, client = client_data)
            if reactions and bkhotel.count() > 0:
                bkcity.bookmark = True
            else:
                bkcity.bookmark = False
    return render(request, "taaplanner/cityreactions.html",
                  {"cities": city_reactions })

def hotelreactionspageview(request, hotelslug):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    try:
        hotel_data = Hotel.objects.get(slug = hotelslug)
    except Hotel.DoesNotExist:
        return render(request, "taaplanner/hotelreactions.html",
                      {"hotels": [] })

```

```

        hotel_reactions = Hotelreaction.objects.filter(hotel =
hotel_data)
        hotel_reactions = hotel_reactions.select_related("hotel")
        if client_data:
            for bkhotel in hotel_reactions:
                reactions = Hotelreaction.objects.filter(hotel =
bkhotel, client = client_data)
                if reactions and reactions.count() > 0:
                    bkhotel.reaction = True
                else:
                    bkhotel.reaction = False
                bookmarks = Hotelbookmark.objects.filter(hotel =
bkhotel, client = client_data)
                if reactions and bkhotel.count() > 0:
                    bkhotel.bookmark = True
                else:
                    bkhotel.bookmark = False
            return render(request, "taaplanner/hotelreactions.html",
{"hotels": hotel_reactions })

def packagereactionspageview(request, packageslug):
    try:
        client_data = Client.objects.get(user = request.user)
    except Client.DoesNotExist:
        client_data = None
    try:
        travelpackage_data = Travelpackage.objects.get(slug =
packageslug)
    except Travelpackage.DoesNotExist:
        return render(request,
"taaplanner/packagereactions.html",
{"packages": [] })
    package_reactions =
Travelpackagereaction.objects.filter(client = reaction_client_data)
    package_reactions =
package_reactions.select_related("travelpackage")
    if client_data:
        for bkpackage in package_reactions:
            reactions =
Travelpackagereaction.objects.filter(travelpackage = bkpackage, client
= client_data)
            if reactions and reactions.count() > 0:
                bkpackage.reaction = True
            else:
                bkpackage.reaction = False
            bookmarks =
Travelpackagebookmark.objects.filter(travelpackage = bkpackage, client
= client_data)
            if reactions and reactions.count() > 0:

```

```
        bkpackage.bookmark = True
    else:
        bkpackage.bookmark = False
    return render(request, "taaplanner/clientreactions.html",
                  {"packages": package_reactions})
```

ДОДАТОК В
Слайди презентації

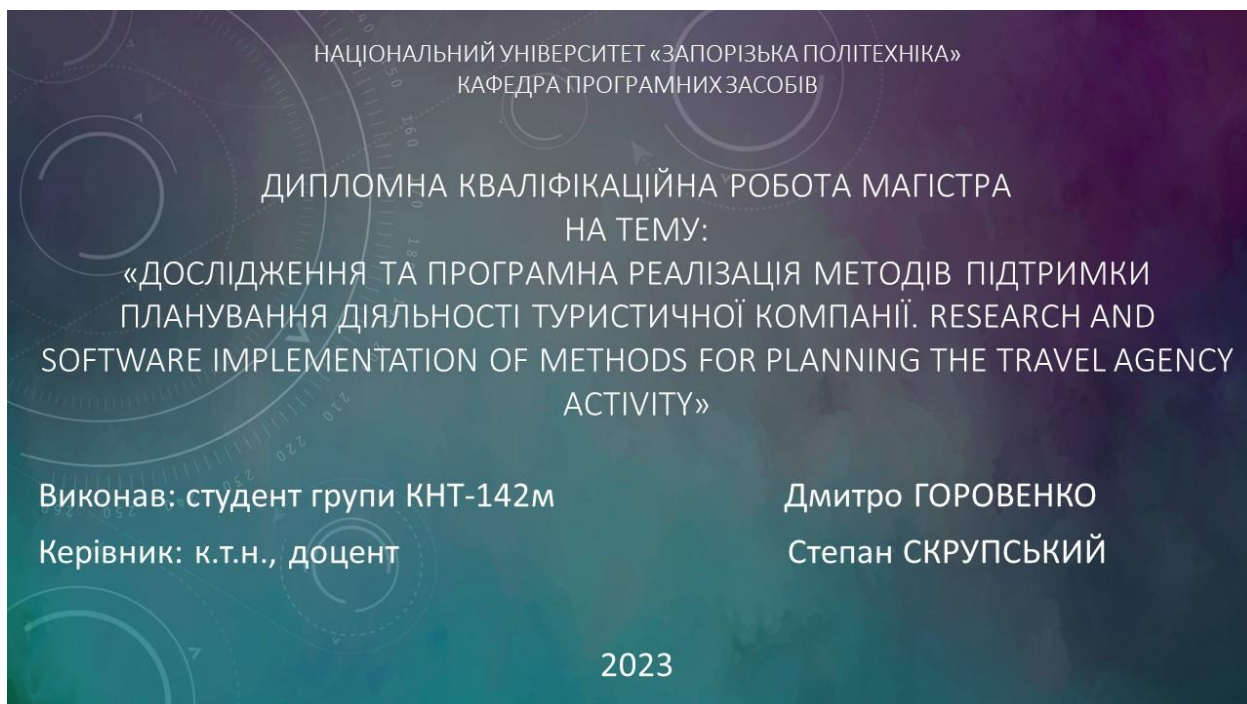


Рисунок В.1 – Титульний слайд

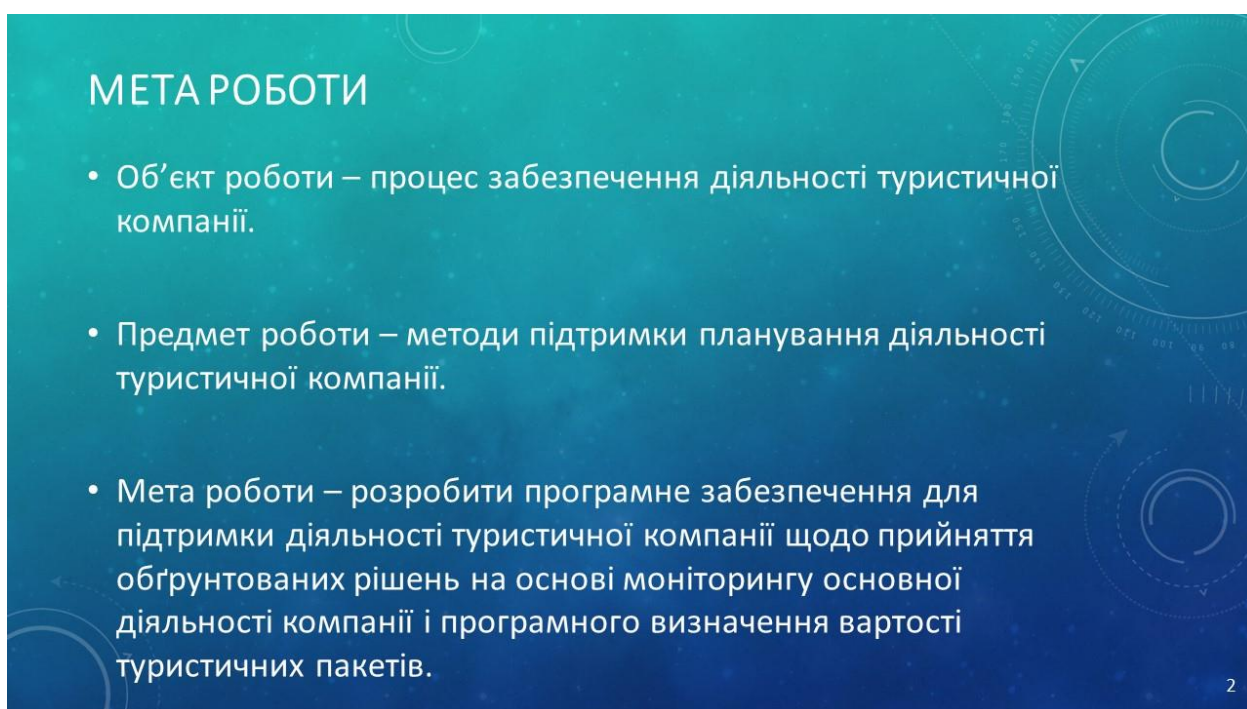


Рисунок В.2 – Об'єкт, предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- аналіз програм, які підтримують роботу туристичної компанії з клієнтами;
- дослідження методів і моделей визначення вартості туристичного пакета;
- проектування програмного забезпечення підтримки діяльності туристичної компанії;
- реалізація програми підтримки діяльності туристичної компанії.

3

Рисунок В.3 – Завдання роботи

ВИЗНАЧЕННЯ ОСОБЛИВОСТЕЙ РОЗРОБЛЮВАНОЇ ТА ІСНУЮЧИХ ПРОГРАМ

Критерій	TUI	Viator	Розробка
Засоби пошуку	Туристичні пакети, рейси	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі
Засоби збереження	Туристичні пакети	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі, рейси
Замовлення туристичних пакетів	Так	Так	Так
Засоби реагування	Ні	Туристичні пакети, екскурсії	Туристичні пакети, міста, країни, готелі
Визначення вартості туристичного пакета на основі прогнозування	Ні	Ні	Так

4

Рисунок В.4 – Визначення особливостей розроблюваної та існуючих програм

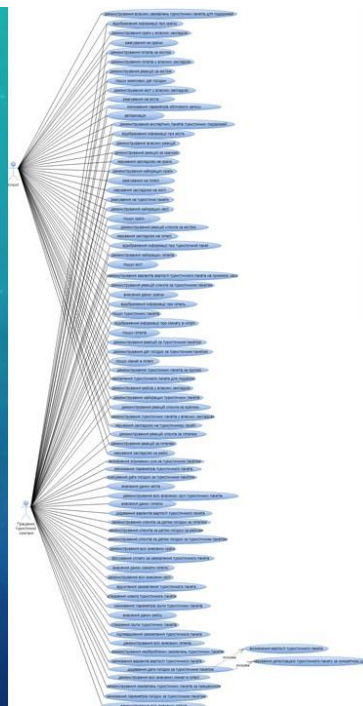
МЕТОД ВИЗНАЧЕННЯ ВАРТОСТІ ТУРИСТИЧНОГО ПАКЕТА

- виділення значень характеристик туристичного пакета для формування вибірки;
- обчислення і кодування значень характеристик екземплярів навчальної вибірки даних, включаючи зокрема визначення святкових днів, основного міста подорожі;
- модель визначення вартості туристичного пакета має бути створена на основі ансамбля дерев рішень за допомогою її подальшого навчання на основі екстремального градієнтного підсилювання, реалізуючи навчання моделі на основі вибірки даних, що було сформовано;
- використання навченої моделі для визначення вартості нового туристичного пакета, який повинен бути охарактеризований та перетворений на відповідний екземпляр вибірки даних.

5

Рисунок В.5 – Метод визначення вартості туристичного пакета

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ ПРОГРАМИ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ТУРИСТИЧНОЇ КОМПАНІЇ



6

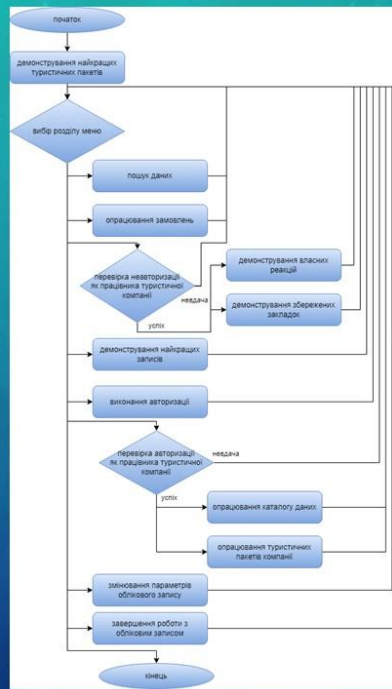
Рисунок В.6 – Діаграма варіантів використання програми для підтримки діяльності туристичної компанії

СХЕМА БАЗИ ДАНИХ



Рисунок В.8 – Схема бази даних

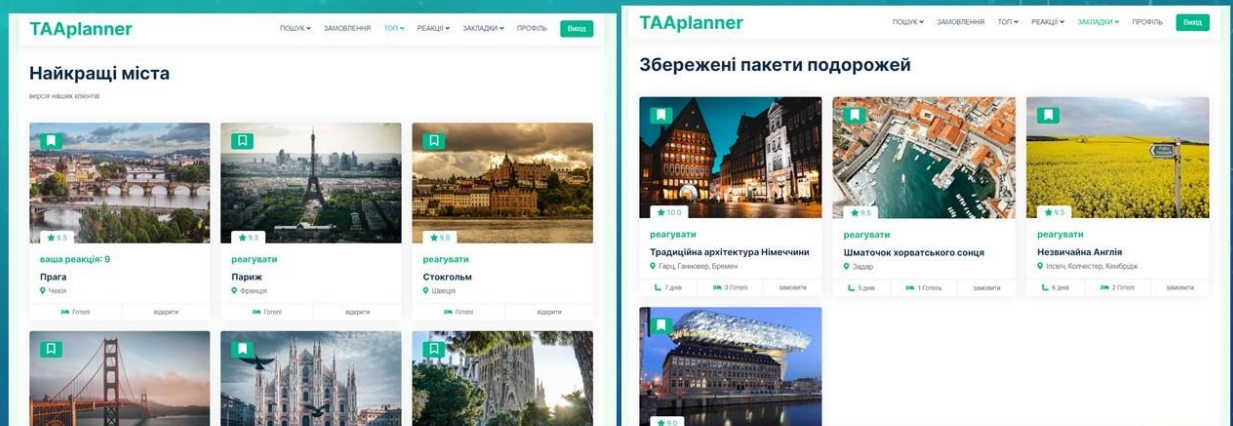
ФУНКЦІОНАЛЬНА СХЕМА ПРОГРАМИ ПІДТРИМКИ ДІЯЛЬНОСТІ ТУРИСТИЧНОЇ КОМПАНІЇ



9

Рисунок В.9 – Функціональна схема програми підтримки діяльності туристичної компанії

РОБОТА КЛІЄНТА З ПРОГРАМОЮ



10

Рисунок В.10 – Робота клієнта з програмою

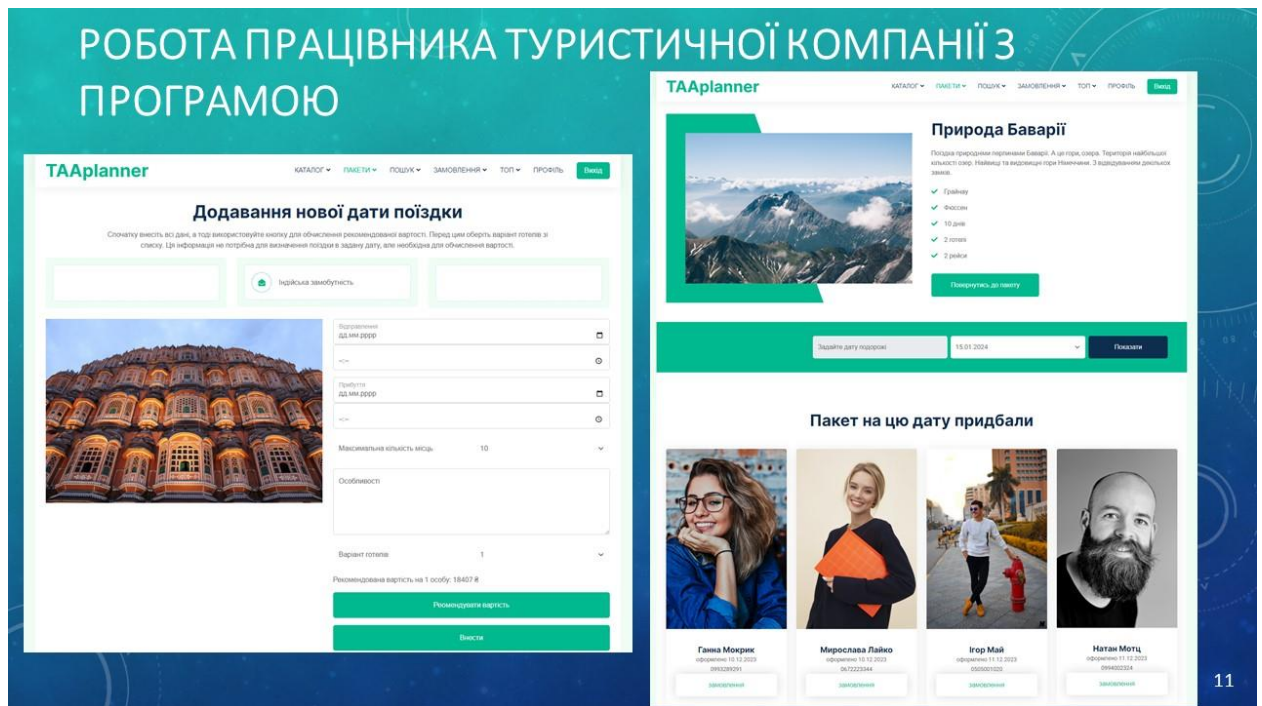


Рисунок В.11 – Робота працівника туристичної компанії з програмою

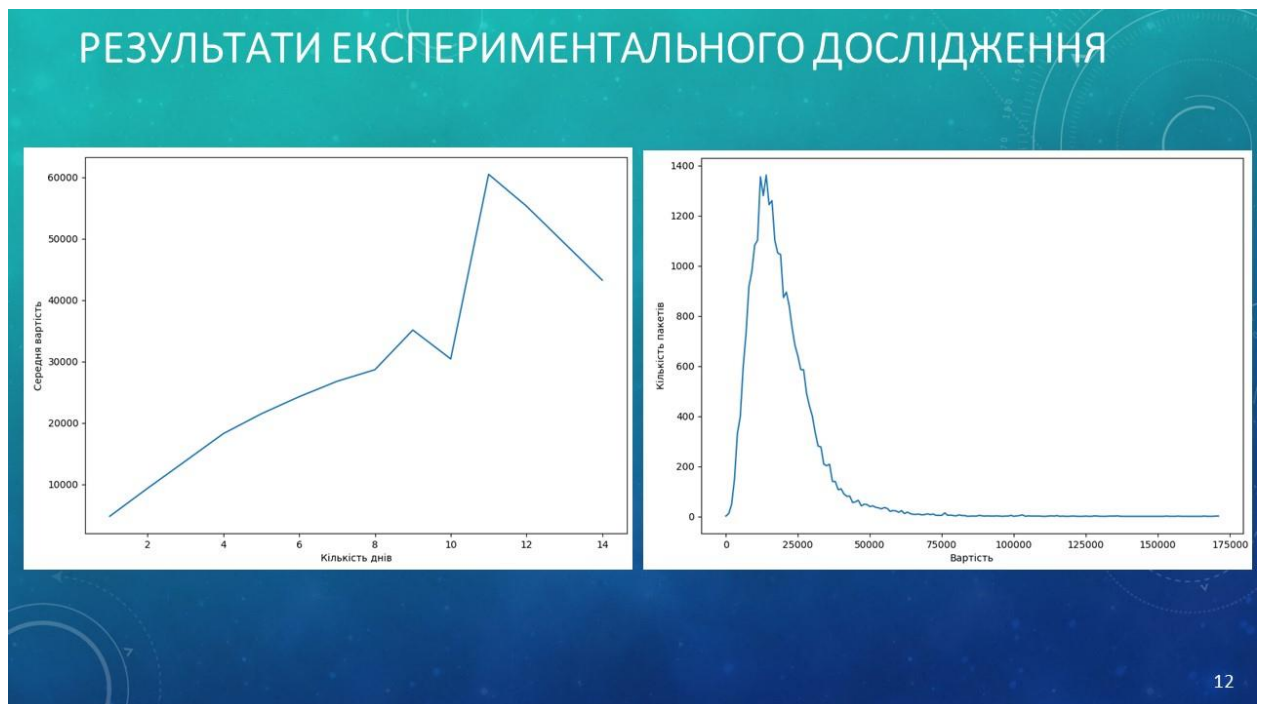


Рисунок В.12 – Результати експериментального дослідження

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

Кількість дерев у ансамблі	MAE	Коефіцієнт детермінації	Модель	MAE	Коефіцієнт детермінації
10	0,0202002	0,7692323	Лінійна регресія	0,0343441	0,4422857
20	0,0182589	0,7916033	Дерево рішень	0,0239499	0,6733814
40	0,0172807	0,8031989	Екстремальне градієнтне підсилювання для ансамблів дерев рішень	0,0164359	0,806069
50	0,0170289	0,8043945			
100	0,0164359	0,806069			
400	0,0167106	0,7899235			
500	0,0168415	0,7864779			

13

Рисунок В.13 – Результати експериментального дослідження

ВИСНОВКИ

Програмне забезпечення для підтримки діяльності туристичної компанії базується на використанні вбудованої моделі, яку було створено і навчено засобами програми і яка дозволяє виконувати визначення вартості туристичного пакету, що дозволяє разом з засобами моніторингу і аналізу основної діяльності компанії та її клієнтів приймати обґрунтовані управлінські рішення в підсумку.

Наукова новизна роботи полягає в запропонованому методі визначення вартості туристичного пакета, який визначає обчислення вхідних ознак моделі на основі характеристик туристичного пакета, виконує створення моделі на основі екстремального градієнтного підсилювання для навчання ансамблів дерев рішень.

Результати експериментального дослідження показали, що інформативність створеної на основі методу визначення вартості туристичного пакета моделі є значно вищою порівняно з іншими дослідженими моделями, що разом з покращеними значеннями MAE говорить про можливість використання на практиці створеної моделі та загалом перспективні отримані значення. Результуюча модель мала значення MAE на 31,4 % краще за модель на основі одного окремого дерева рішень та на 52,1 % краще за лінійну регресію.

14

Рисунок В.14 – Висновки